UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

AJUSTES DE PARÁMETROS DE LA JVM PARA UN MEJOR DESEMPEÑO DE CMM

INFORME DE ACTIVIDADES

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

ARMANDO RODRIGUEZ MALDONADO

ASESOR DE INFORME:

M.C. ALEJANDRO VELÁZQUEZ MENA

CIUDAD UNIVERSITARIA 2012

Agradecimientos

Estoy agradecido con la Facultad de Ingeniería de la Universidad Nacional Autónoma de México por darme la oportunidad de tener una educación profesional y cobijarme con su gran amor para convertirme en el hombre que ahora soy.

Agradezco a todos mis profesores de carrera que día a día sembraron en mi la tenacidad para convertirme en el profesionista que ahora soy.

Y finalmente agradezco a la persona mas hermosa del mundo que cambio mi forma de pensar y existir, que le dio sentido a mi andar por la vida desde que escuche su primer llanto, que con cada sonrisa me dice “supérate papito”, que tan solo con ver una imagen mía me entrega su amor de forma incondicional, inocente, puro y sincero, que tan solo con un beso me arrodillo a sus pies para decirle “por ti cariño soy y seré el mejor papá del mundo”. A ti Liam Michelle este logro es una correspondencia a la felicidad por haberme escogido como tu papá.

Mi corazón algún día dejará de latir pero mi amor por mi Universidad y mi bebita persistirán para la eternidad.

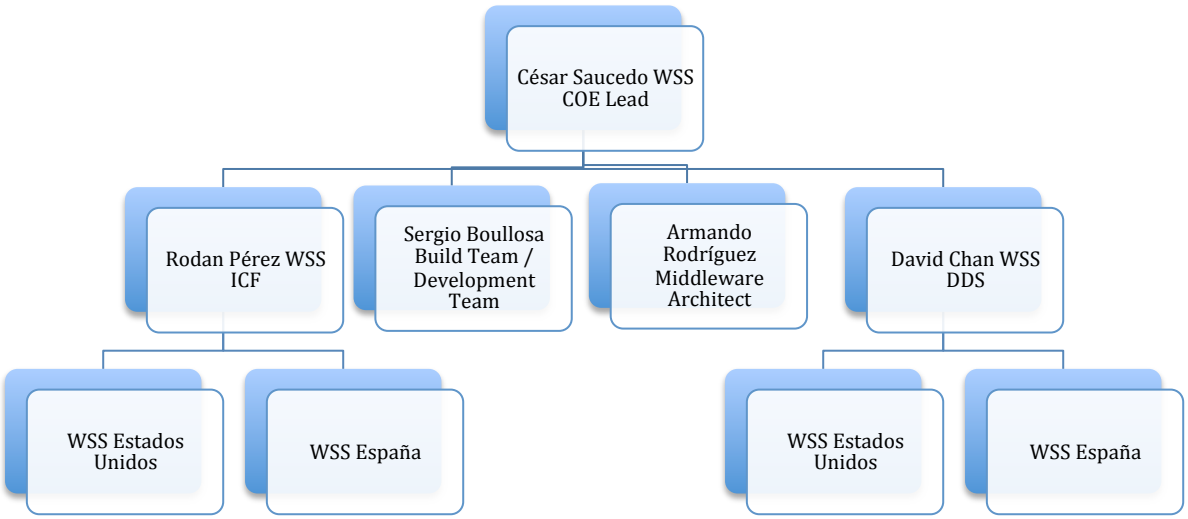
CAPITULO I

Organización de la empresa

Softtek

Softtek fué fundada en 1982, es un proveedor global de servicios orientados a procesos de TI con más de 6,000 colaboradores en 30 oficinas en Norteamérica, Latinoamérica, Europa y Asia. Con nueve Centros de Desarrollo Global en México, China, Brasil, Argentina y España, Softtek mejora el tiempo de entrega de soluciones de negocio, reduce costo de las aplicaciones existentes, entrega aplicaciones mejor diseñadas y probadas, y produce resultados predecibles para grandes empresas en más de 20 países. A través de modelos de servicios de entrega on-site, on-shore y su marca registrada Global Nearshore, Softtek ayuda a los CIOs a incrementar el alineamiento con el negocio. Softtek es el creador y líder de la industria nearshore.

1.1 Organigrama de Softtek



1.2 Cánones de Softtek

La vida organizacional de Softtek es bajo la convicción de que requieren ser asimilados de manera personal.

El accionar como equipo bajo los mismos cánones permite que la organización evolucione.

Apertura. Soy abierto cuando interactúo con las personas sin retener ni distorsionar lo que pienso y siento. Cuanto más consciente soy de lo que pasa conmigo, más claramente puedo identificar mis preferencias y temores; de ésta forma genero relaciones más efectivas.

Autodeterminación. Me autodetermino cuando me siento responsable de lo que me sucede y reconozco la capacidad que tengo de elegir cada una de mis acciones, reacciones, sentimientos y autoconcepto. Desde ésta perspectiva, obtengo un poder de transformación sobre mi vida, no culpo a otros y asumo mi responsabilidad para diseñar la persona que quiero ser.

Compromiso. Soy comprometido cuando cumplo todas las promesas que hago, encargándome de hacer lo necesario para que ocurra lo que prometí.

Confianza. Confío en las personas cuando creo que actuarán de buena fe. Cuando confío en una persona, le doy la oportunidad de creer en sí misma y superarse; así obtengo mayor tranquilidad y fortaleza en nuestra relación. La confianza se otorga, no se gana.

Entusiasmo. Soy entusiasta cuando estoy en búsqueda de retos continuos, cada vez más ambiciosos. Me entusiasmo cuando enfrento un desafío que me hace evolucionar, tomar riesgos, superar mis temores y contagiar a todo aquel que se identifique conmigo.

Flexibilidad. Soy flexible cuando me adapto a diferentes ambientes y personas. Ser flexible me da la posibilidad de dialogar, crear alternativas y adecuarlas al contexto presente.

Perseverancia. Soy perseverante cuando estoy seguro de que alcanzaré mis metas y tengo la energía para permanecer en el empeño hasta que tarde o temprano lo consiga.

Respeto. Respeto a otras personas cuando me relaciono con ellas de forma efectiva, aún cuando tengan creencias, preferencias y comportamientos diferentes a los míos. En la medida en que las respeto, dejo de imponer mis puntos de vista y aprendo más de las distintas creencias y percepciones que existen a mi alrededor.

Sinergia. Logro sinergia cuando uno mis esfuerzos a los de un grupo para multiplicar nuestras fortalezas y alcanzar grandes metas, más allá de la suma de los esfuerzos individuales.

Sociedad. Soy socio cuando mi visión personal coincide con la de Softtek y actúo como dueño y parte de la organización, buscando crear un patrimonio sólido, a través de compartirlo y multiplicarlo con otras personas. Ser socio me transforma en emprendedor y

visionario, apostando e invirtiendo en Softtek con la firme idea de trascender.

Visión. Construyo mi visión cuando identifico mis sueños y estoy dispuesto a entregar lo mejor de mí para hacerlos realidad. Cuando soy claro en mi visión y mis acciones, trasciendo, encuentro la fuerza para movilizarme, supero la adversidad y contribuyo de forma positiva a la sociedad.

CAPITULO II

Participación en Proyectos

Tesorería – Softtek - Actual

Actualmente soy el responsable del desempeño y de la arquitectura de las siguientes aplicaciones web:

CMM (del inglés Cash Managment Module).- Aplicación cuyo objetivo principal es el flujo de efectivo, actualización de tasas de interés a nivel mundial, cálculo de préstamos bancarios.

IDM (del inglés Identity Data Module).- Aplicación web cuya función principal es la de levantar requerimientos de los usuarios para accesos a las cajas de Unix, Windows NT, Citrix y Bases de Datos.

TrinityGics.- cuyo sitio es www.trinitygics.com ésta aplicación sirve básicamente para ofrecer al cliente tipos de fondo para inversión a los municipios, autoridades estatales y locales y otras partes que tienen que invertir producto de los bonos y otros fondos.

HPSM (del inglés Hewlett Packard Service Manager).- Es un software escalable y robusto cuyo objetivo es dar solución a incidentes, normalizar gestión de procesos, ofrecer calidad en la prestación de servicios y soporte al usuario final. Proporciona un centro de comunicación único, al Gerente de Servicio le permite trabajar como una sola organización regida por un conjunto coherente de los procesos. Su robusta funcionalidad se basa en construir en las mejores prácticas ITIL (del inglés Information Technology Infrastructure Library). A su vez también disminuye el tiempo de valoración, Ofrece alta efectividad de los procesos y eficiencia en la reducción de costos, controla el proceso de normalización, mitiga los riesgos y asegurar el cumplimiento de solicitudes automatizadas con el usuario final.

También hago nuevos desarrollos, mejoras de código, pruebas unitarias, e integrales y su documentación correspondiente.

Nafinsa

Fui el Líder Técnico Java y responsable del portal www.nafinsa.com . Diseñé, desarrollé e implementé los módulos de alta, baja, cambios y búsqueda del sistema SIE (Sistema de Integración Empresarial) perteneciente a dicho portal.

Grupo Salinas IUSACELL

Fui Administrador de Websphere y Desarrollador Java Senior, fui el responsable de las siguientes liberaciones:
Banco Azteca.- Red Móvil Azteca
Iusacell.- Ubicacel Familiar para BlackBerry

Elektra.- Modulo de autenticación al portal.

Desarrollé una aplicación web para el monitoreo de bases de datos de producción y de aplicaciones web.

Metlife - Hildebrando

Fui líder de proyecto del equipo pruebas bajo la metodología T-Process (del inglés Test Process) implementada por Hildebrando. Dicha metodología está basada en la ejecución de pruebas unitarias, pruebas integrales, pruebas de caja blanca y pruebas caja negra. También fui el encargado de la entrega de la documentación generada de dichas pruebas.

Pagatodo - Hildebrando

Desarrollé un módulo cuya función era garantizar la recarga de saldo en los teléfonos celulares en fechas de promoción.

BOLSA MEXICANA DE VALORES – Hildebrando

Desarrollé el módulo de una gráfica que hacia los balances del comportamiento de la Bolsa Mexicana de Valores en tiempo real

SCOTIA BANK – Hildebrando

Desarrollé los módulos de alta, baja, actualización y búsqueda de la aplicación web SUBASTAS cuyo sitio es https://subastas.scotiainlatrade.com/SubastasAppWeb/login.jsp el objetivo principal era la gestión de los clientes, acreedores y manejo de efectivo.

CAPITULO III

Desempeño de la JVM de CMM

3.1.0 Introducción

La máquina virtual de java de proceso nativo, es ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial, el cual es generado por el compilador del lenguaje Java.

La JVM es una de las piezas fundamentales de la plataforma Java. Básicamente se sitúa en un nivel superior al Hardware del sistema sobre el que se pretende ejecutar la aplicación, y éste actúa como un *puente* que entiende tanto el bytecode, como el sistema sobre el que se pretende ejecutar. Así, cuando se escribe una aplicación Java, se hace pensando que será ejecutada en una máquina virtual Java en concreto, siendo ésta la que en última instancia convierte de código bytecode a código nativo del dispositivo final.

La gran ventaja de la máquina virtual java es aportar portabilidad al lenguaje de manera que desde Sun Microsystems se han creado diferentes máquinas virtuales java para diferentes arquitecturas y así un programa con extensión .class escrito en un Windows puede ser interpretado en un entorno Linux, donde es solo es necesario disponer de dicha máquina virtual para dichos entornos. De ahí el famoso axioma que sigue a Java, "escríbelo una vez, ejecútalo en cualquier parte", o "Write once, run anywhere".

Pero, los intentos de la compañía propietaria de Java y productos derivados de construir microprocesadores que aceptaran el Java bytecode como su lenguaje de máquina fueron más bien infructuosos.

La máquina virtual de Java puede estar implementada en software, hardware, una herramienta de desarrollo o un Web browser; lee y ejecuta código precompilado que es independiente de la plataforma multiplataforma. La JVM provee definiciones para un conjunto de instrucciones, un conjunto de registros, un formato para archivos de clases, pila, un heap con recolector de basura y un área de memoria. Cualquier implementación de la JVM que sea aprobada por SUN debe ser capaz de ejecutar cualquier clase que cumpla con la especificación.

Existen varias versiones, en orden cronológico, de la máquina virtual de Java. En general la definición del Java bytecode no cambia significativamente entre versiones, y si lo hacen, los desarrolladores del lenguaje procuran que exista compatibilidad hacia atrás con los productos anteriores.

A partir de J2SE 5.0, los cambios en la especificación de la JVM han sido desarrollados bajo el auspicio de la JCP (del inglés Java Community Process) y especificada en la JSR 924. Desde el año 2006, cambios en la especificación para soportar las modificaciones del formato del fichero de clases se están llevando a cabo en una versión de mantenimiento en la JSR 924.

Kaffe es un ejemplo de una implementación de JVM desde cero. Sun es la propietaria de la marca registrada "Java", que usa para certificar aquellas implementaciones que se ajustan y son totalmente compatibles con sus especificaciones.

Para poder ejecutar una aplicación en una Máquina Virtual de Java, el programa código debe compilarse de acuerdo a un formato binario portable estandarizado, normalmente en forma de ficheros con extensión .class. Un programa puede componerse de múltiples clases, en cuyo caso cada clase tendrá asociada su propio archivo .class. Para facilitar la distribución de aplicaciones, los archivos de clase pueden empaquetarse juntos en un archivo con formato jar. Ésta idea apareció en la época de los primeros applets de Java. Estas aplicaciones pueden descargar aquellos archivos de clase que necesitan en tiempo de ejecución, lo que suponía una sobrecarga considerable para la red en una época donde la velocidad suponía un problema. El empaquetado evita la sobrecarga por la continua apertura y cierre de conexiones para cada uno de los fragmentos necesarios.

El código resultante de la compilación es ejecutado por la JVM que lleva a cabo la emulación del conjunto de instrucciones, bien por un proceso de interpretación o más habitualmente mediante un compilador JIT (del inglés Just In Time), como el HotSpot de Sun. Ésta última opción convierte el bytecode a código nativo de la plataforma destino, lo que permite una ejecución mucho más rápida. El inconveniente es el tiempo necesario al principio para la compilación.

En un sentido amplio, la Máquina Virtual de Java actúa como un puente entre el resultado de la compilación y el sistema sobre el que se ejecuta la aplicación. Para cada dispositivo debe haber una JVM específica, ya sea un teléfono móvil, un PC con Windows XP, o un microondas. En cualquier caso, cada máquina virtual conoce el conjunto de instrucciones de la plataforma destino, y traduce un código escrito en lenguaje Java (común para todas) al código nativo que es capaz de entender el Hardware de la plataforma.
La JVM *verifica* todo bytecode antes de ejecutarlo. Esto significa que solo una cantidad limitada de secuencias de bytecode forman programas válidos, por ejemplo una instrucción JUMP (branch) puede apuntar solo a una instrucción dentro de la misma función. A causa de esto, el hecho de que JVM es una arquitectura de pila no implica una carga en la velocidad para emulación sobre arquitecturas basadas en registros cuando usamos un compilador JIT: no hay diferencia para un compilador JIT si nombra registros con nombres imaginarios o posiciones de pila imaginarias que necesitan ser ubicadas a los registros de la arquitectura objetivo. De hecho, la verificación de código hace a la JVM diferente de una arquitectura clásica de pila cuya emulación eficiente con un compilador JIT es más complicada y típicamente realizado por un intérprete más lento.

La verificación de código también asegura que los patrones de bits arbitrarios no pueden usarse como direcciones. La protección de memoria se consigue sin necesidad de una unidad de Gestión de Memoria (del inglés Managment Memory Unit). Así, JVM es una forma eficiente de obtener protección de memoria en chips que no tienen MMU.

La JVM tiene instrucciones para los siguientes grupos de tareas

1. Carga y Almacenamiento
2. Aritméticas
3. Conversión de tipos
4. Creación y manipulación de objetos
5. Gestión de pilas (push / pop)
6. Transferencias de Control (branching)
7. Invocación y retorno a Métodos
8. Lanzar excepciones

La clave es la compatibilidad binaria. Cada sistema operativo de un host particular necesita su propia implementación de JVM y runtime. Estas JVMs interpretan el byte code semánticamente de la misma manera, pero la implementación actual puede variar. Más complicado que solo la emulación de bytecode es la implementación compatible y eficiente de las APIs java las cuales tienen que ser mapeadas para cada sistema operativo de host.
Una arquitectura de máquina virtual permite control de granularidad fina sobre las acciones que el código puede hacer dentro de la máquina. Esto está diseñado para permitir ejecución segura de código no confiable desde fuentes remotas, un modelo usado muy famoso son las Java applets. Los applets se ejecutan dentro de una VM incorporada en el navegador del usuario, ejecutando código descargado desde un servidor HTTP remoto. El código remoto se ejecuta en una "sandbox" altamente restringida, la cual está diseñada para proteger al usuario de código erróneo o malicioso. Los publicadores con recursos financieros suficientes pueden conseguir un certificado con el cual hacer applets con firma digital que las caractericen como seguras, dándoles entonces permisos para salir de la sandbox y acceder al sistema de ficheros local y sistema de red, presumiblemente bajo el control del usuario.
La edición J2SE tiene dos implementaciones de la máquina virtual:

- Java HotSpot Client VM: La máquina virtual por defecto, preparada para obtener el máximo rendimiento en la ejecución de aplicaciones en el entorno cliente, por ejemplo, reduciendo al máximo el tiempo de inicio de una aplicación Java.
- Java HotSpot Server VM: Preparada para obtener el máximo rendimiento en la ejecución de aplicaciones en el entorno de los servidores.

3.1.1 Definición del Problema.

CMM presenta dos problemas principales los cuales son:

- No ésta disponible CMM cuando se generan reportes.
- El recolector de basura no reclama la memoria cuando el proceso se esta ejecutando.

Causas del Problema.

- Los equipos de Application Support y de Middleware observaron que cuando el usuario corre los siguientes reportes

    - GE DAILY EVENTS GENERATION
    - Files Imported from CMM
    - Bank Balance Analysis Report

CMM tiene el comportamiento como se muestra en la fig 3.1.0

Características de software de la aplicación Web CMM:

- Servidor Web: jakarta tomcat ver. 5.5.17
- Sistema Operativo: Unix, Windows
- jdk ver. 1.5.0_17
- Plataforma: J2EE
- Herramienta de monitoreo: Visual VM ver 1.2, Jconsole, JProfiler.

    - GE DAILY EVENTS GENERATION
    - Files Imported from CMM
    - Bank Balance Analysis Report

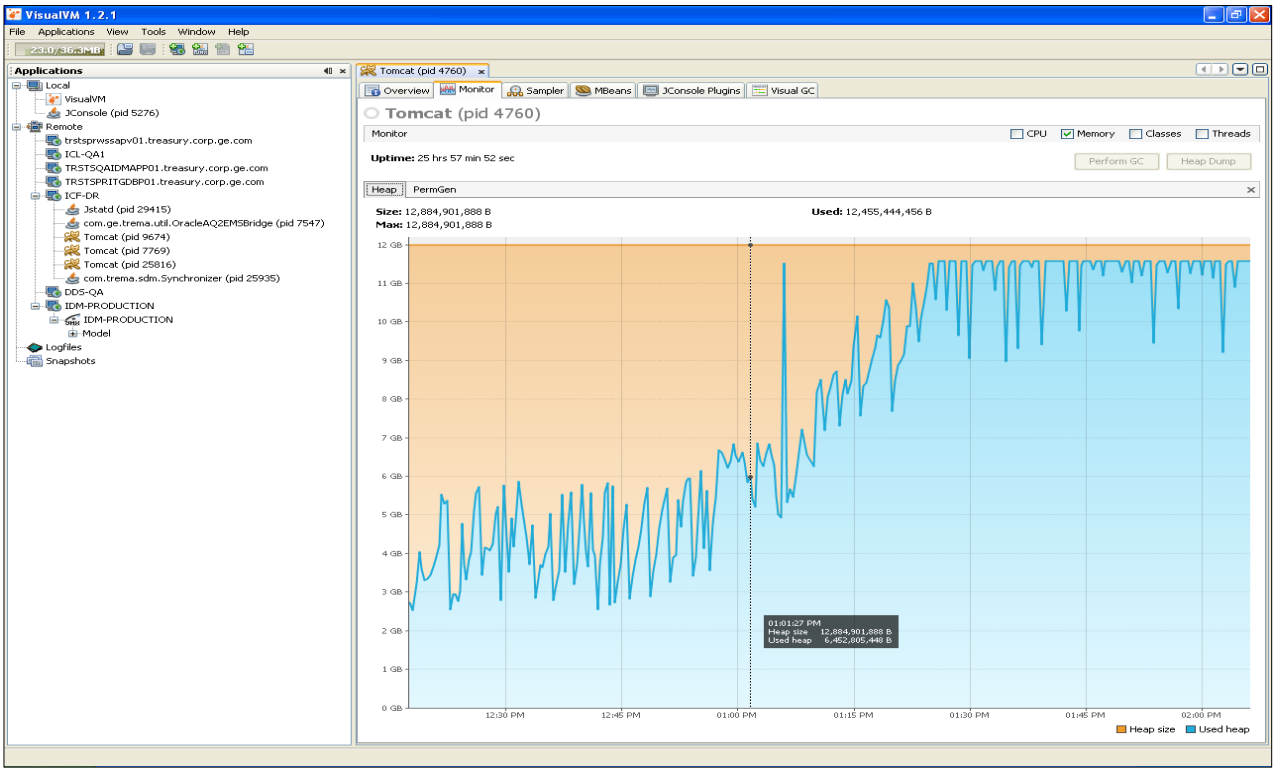


fig. 3.1.0. La aplicación se alenta cuando se corren los reportes antes mencionados.

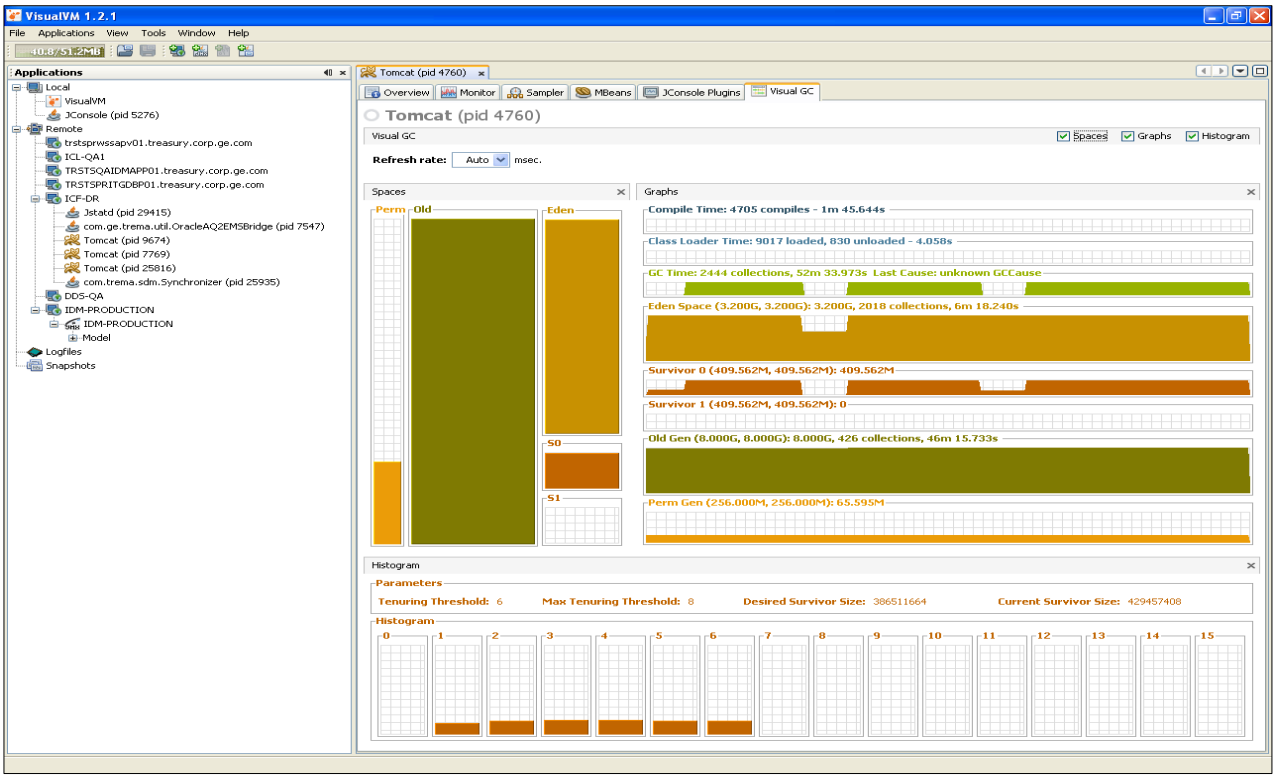


fig. 3.1.1. En esta figura podemos observar que el espacio Permanente de la memoria es menor al 50% mientras que la Old Generacion (del inglés Generación vieja) y el espacio del Eden están saturados.

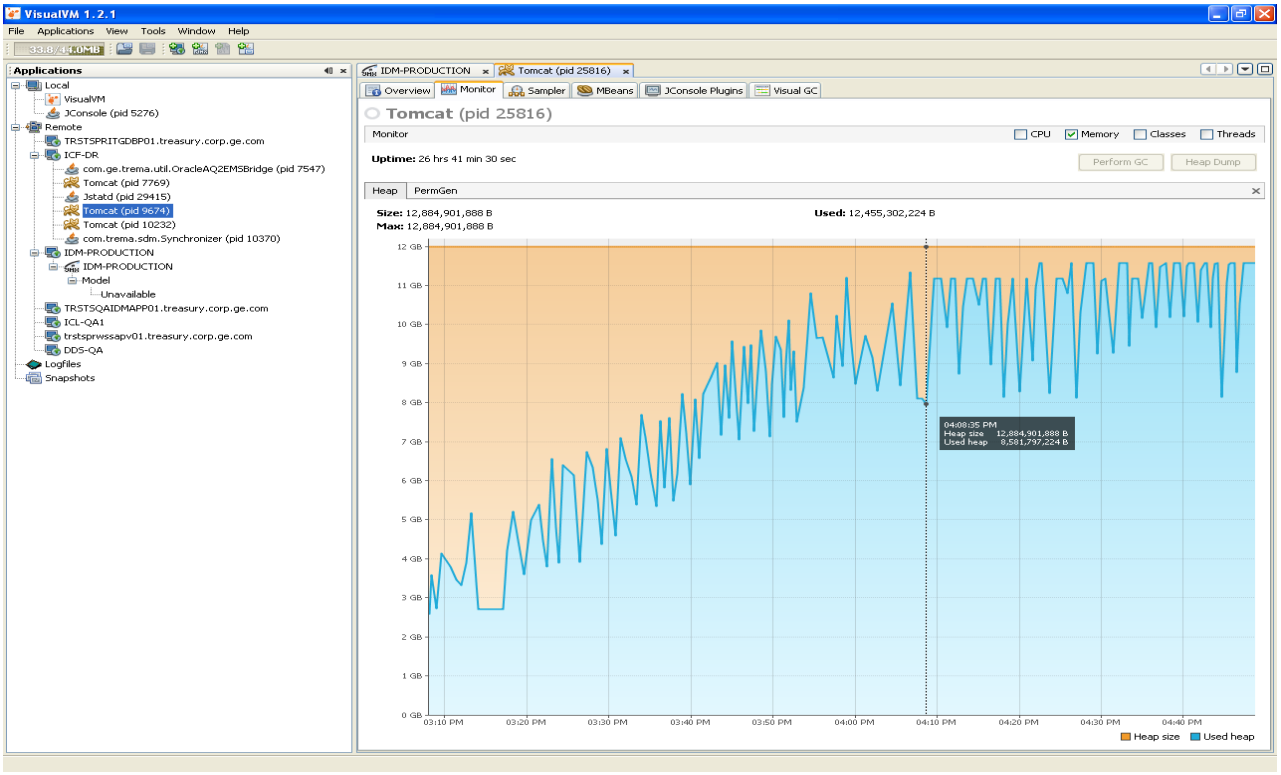


fig. 3.1.2. Después del resteo de la aplicación el problema se vuelve a presentar en la gráfica se observa que se consumen 12.5Gb de 12.8Gb además de que el recolector de basura no reclama la memoria.

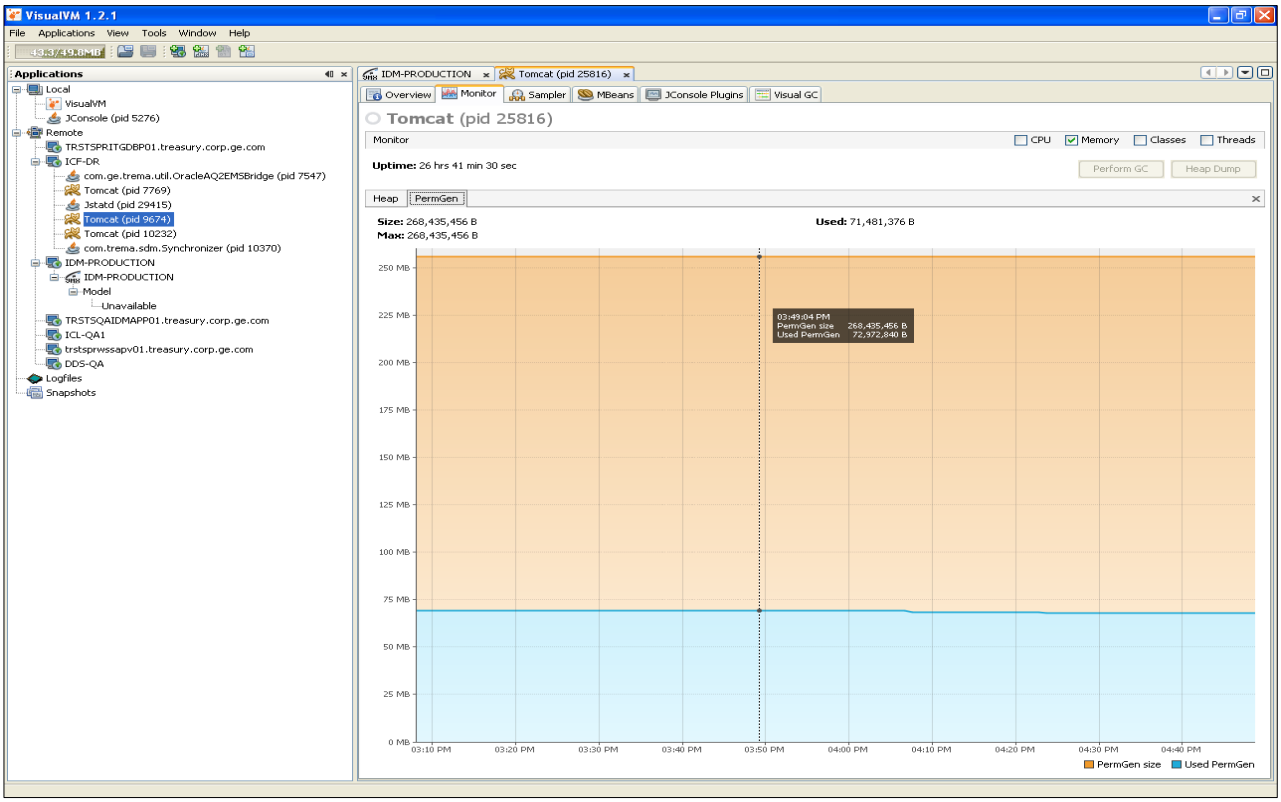


fig. 3.1.3. En esta gráfica se observa que durante la caída de la aplicación la PermGen Size solo requiere menos de 72Mb para cargar las clases y los métodos y este espacio de memoria no es alterado.

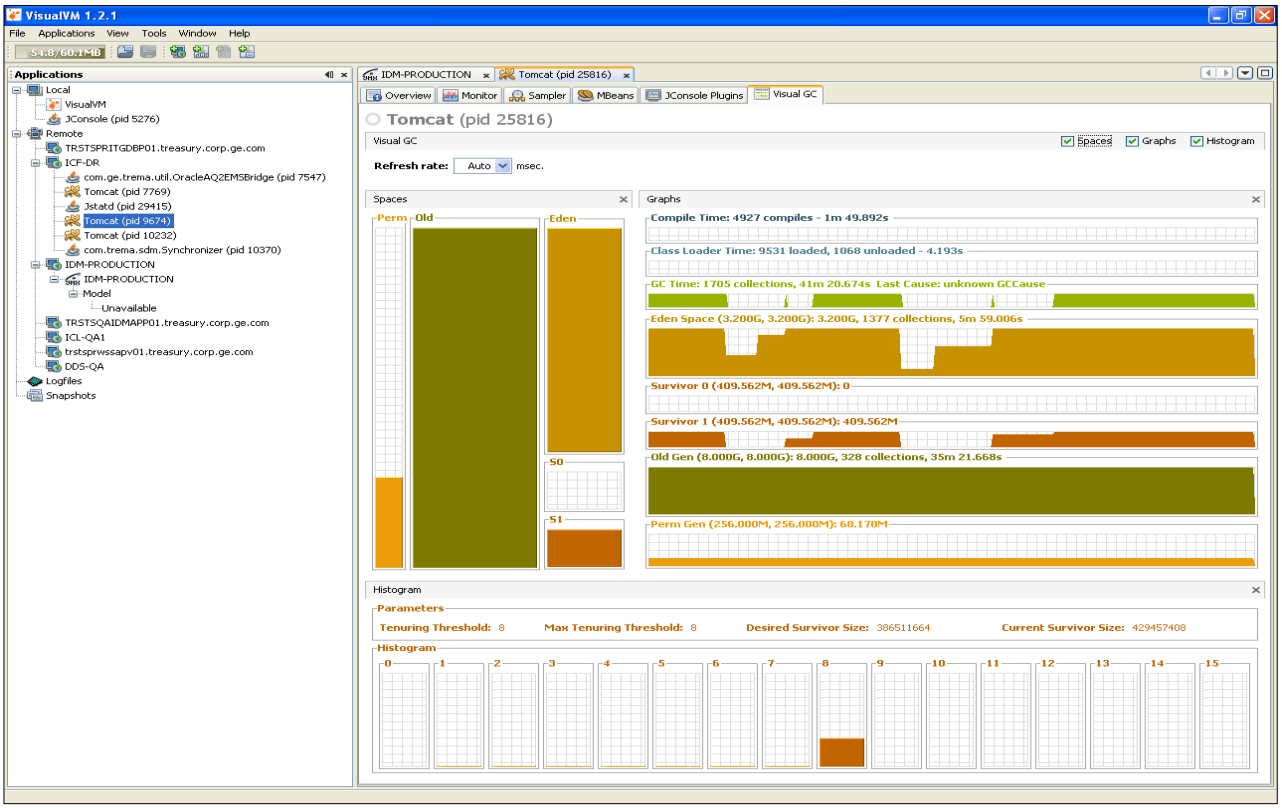


fig. 3.1.4. Nuevamente en esta figura podemos observar que el espacio Permanente de la memoria es menor al 50% mientras que la Old Generacion (del inglés Generación vieja) y el espacio del Eden están saturados.

3.1.2 Requerimiento del cliente

- Hacer un comparativo de herramientas de monitoreo y proponer la mejor para su implementación en la aplicación web.
- Análisis los logs.
- Proponer buenas practicas de desarrollo.
- Análisis la arquitectura de la aplicación web.

Java Visual VM

Java VisualVM es una herramienta que proporciona una interfaz visual para ver la información detallada sobre las aplicaciones Java que se están ejecutando en una Máquina Virtual Java (JVM). Java VisualVM, se proporciona con la distribución de Sun del Java Development Kit (JDK) la mayoría de las herramientas anteriormente independientes JConsole, jstat, jinfo, jstack y jmap forman parte de Java VisualVM. Java VisualVM gestiona estas herramientas para obtener datos desde el software JVM para de ésta forma presentar la información en gráficas. Este monitoreo puede ser local o remoto. En la fig. 3.1.5 podemos observar el desempeño de CMM.

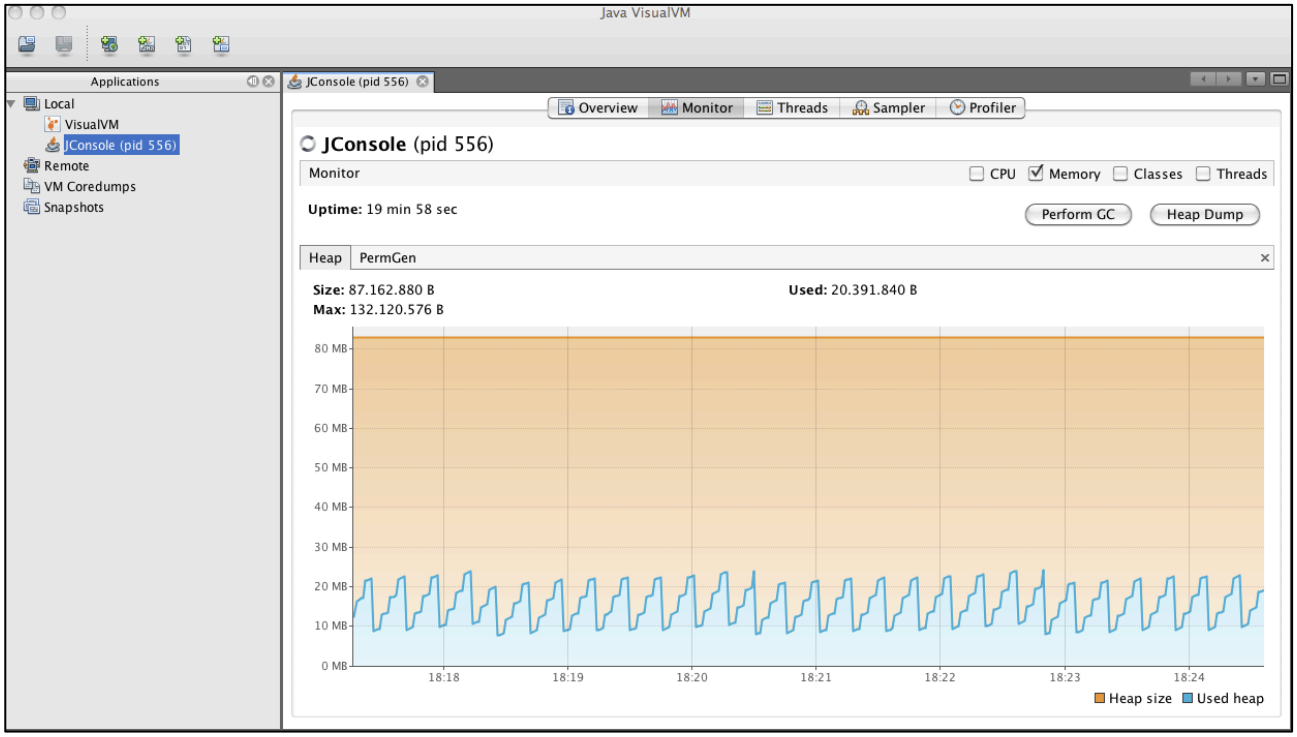


fig. 3.1.5 Monitoreo del uso de la memoria

En la fig. 3.1.6 se observan las instancias de los objetos que están siendo ejecutados desde la máquina virtual (String, Date, HashMap, Float, etc) .

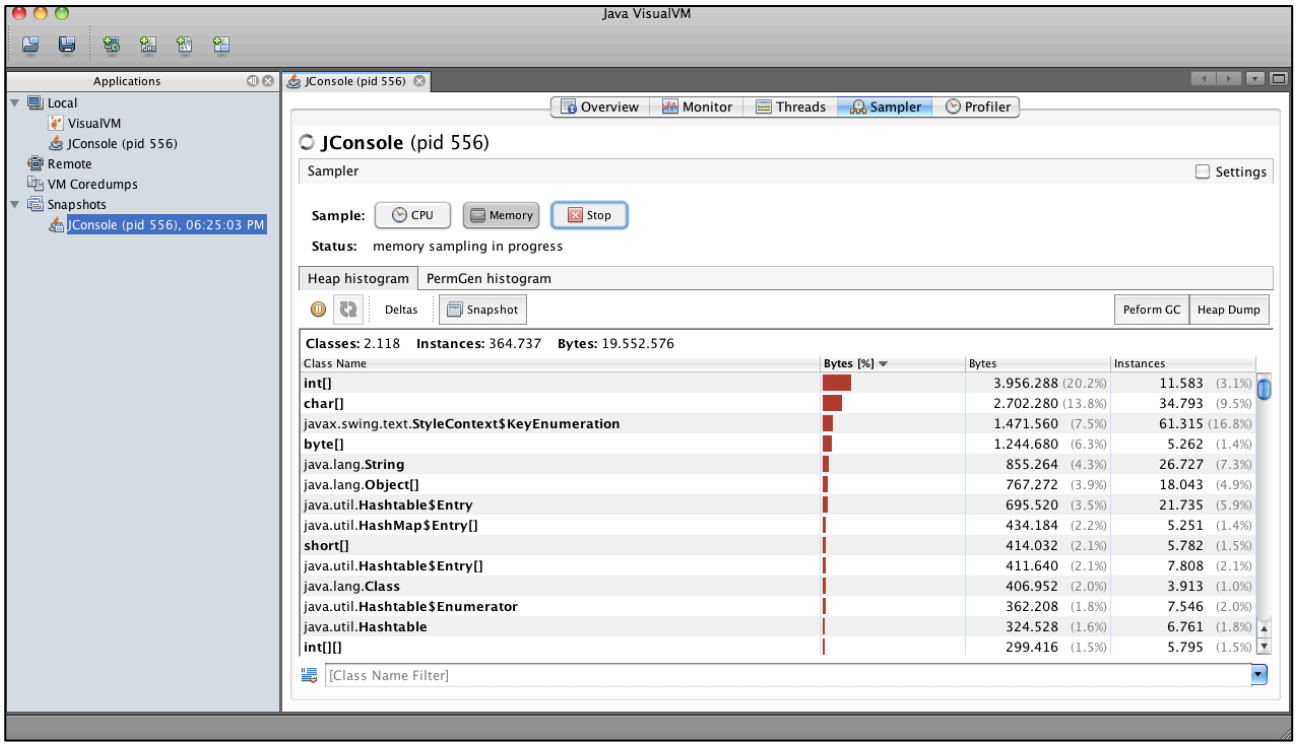


fig. 3.1.6

3.1.3 Monitoreo de aplicaciones java.

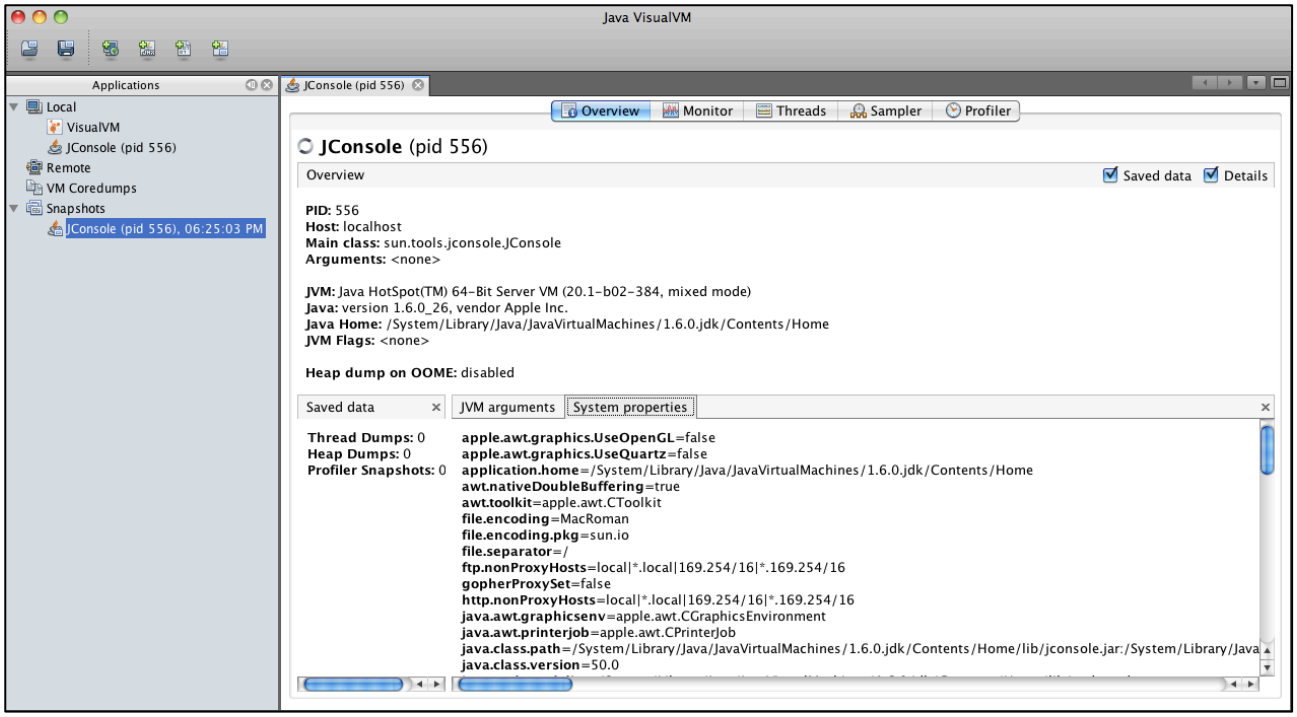


fig. 3.1.7 La aplicación permite no sólo monitorear la JVM en sí, sino también las aplicaciones que estén corriendo dentro de la JVM. Si hacemos doble click sobre él se nos mostrará un panel con información general

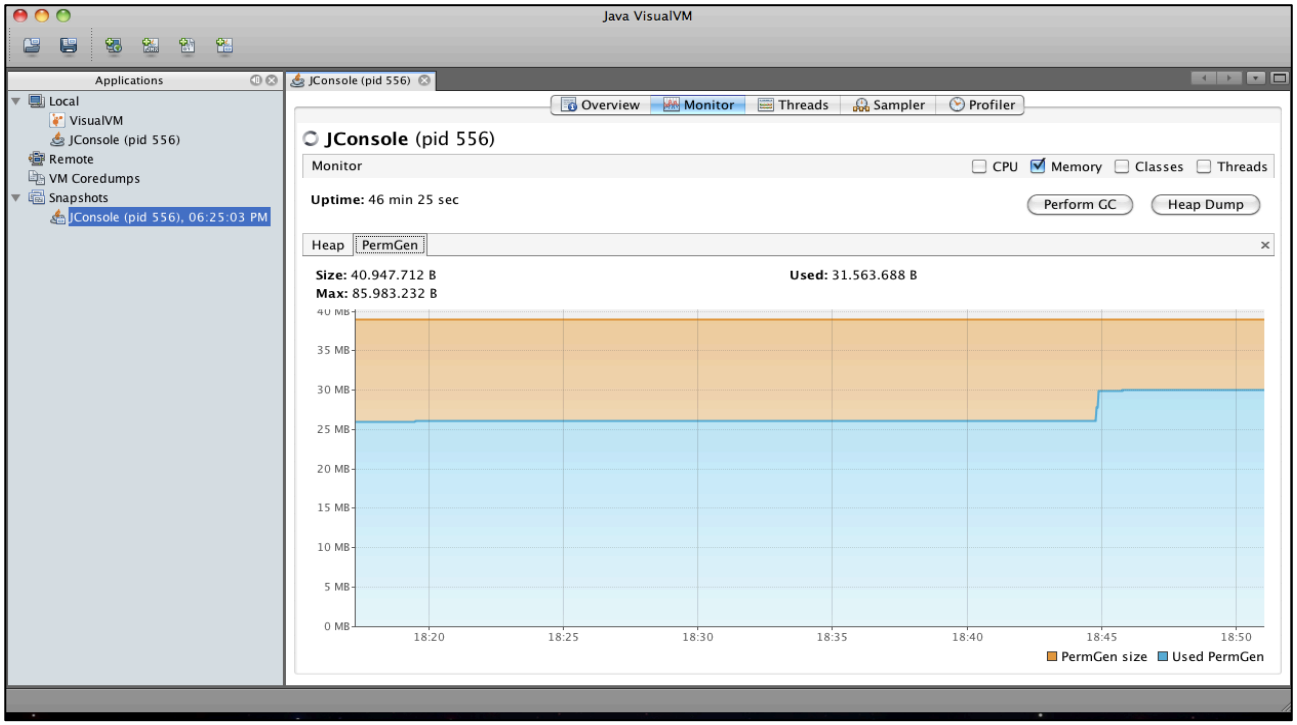


fig. 3.1.8 Al igual que con la JVM, también podemos observar el uso de memoria en la opción Perm Gen de este proceso

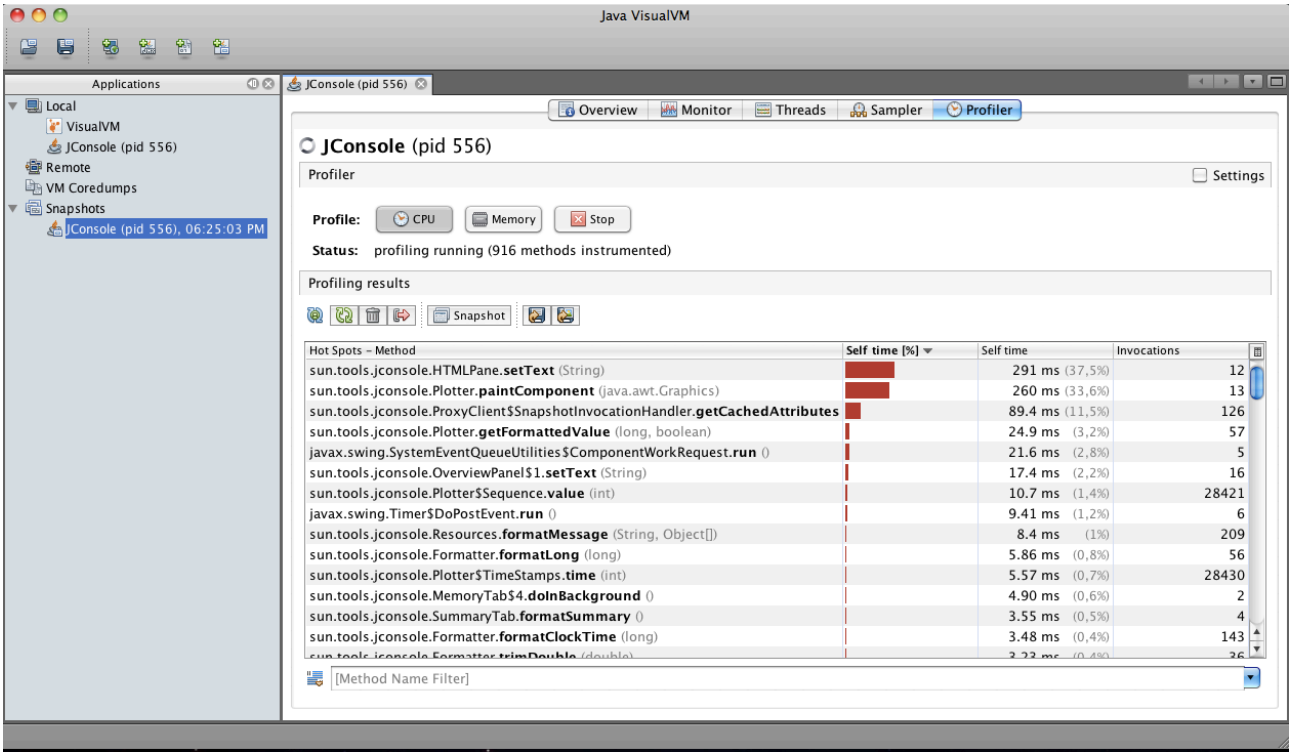


fig. 3.1.9 También podemos buscar posibles cuellos de botella. Para eso hemos de ir al apartado 'Profiler' y escoger qué queremos monitorizar (CPU, memoria, etc.).

También es posible monitorear máquinas virtuales instaladas en otros ordenadores por ejemplo en los servidores de producción.

JConsole

3.1.4 Arquitectura de JConsole

En la Fig.3.2.0 se muestra la arquitectura del monitoreo implementado en la plataforma J2SE 5.0. A partir de ésta versión la JVM (del inglés Java Virtual Machine) es altamente instrumentada para el monitoreo y gestión y ésta a su vez provee información sobre el rendimiento, consumo de recursos.

JMX (Del acrónimo Java Management eXtensions) , provee un estándar para configurar el JRE (java runtime environment). El API (Interfaz de programación de aplicaciones) de JMX permite acceder de forma remota a las aplicaciones. La instrumentación es accesible a través de las interfaces Managed Beans (MBean) de JMX, las cuales están registradas en la plataforma MBean del servidor. Las aplicaciones a su vez también pueden tener sus propios MBeans y regístralos en la plataforma MBean del servidor los cuales a su vez funcionan como punto simple de accesos remotos. Jconsole puede conectarse usando la plataforma MBean y gestionar la aplicación usando la tecnología JMX.

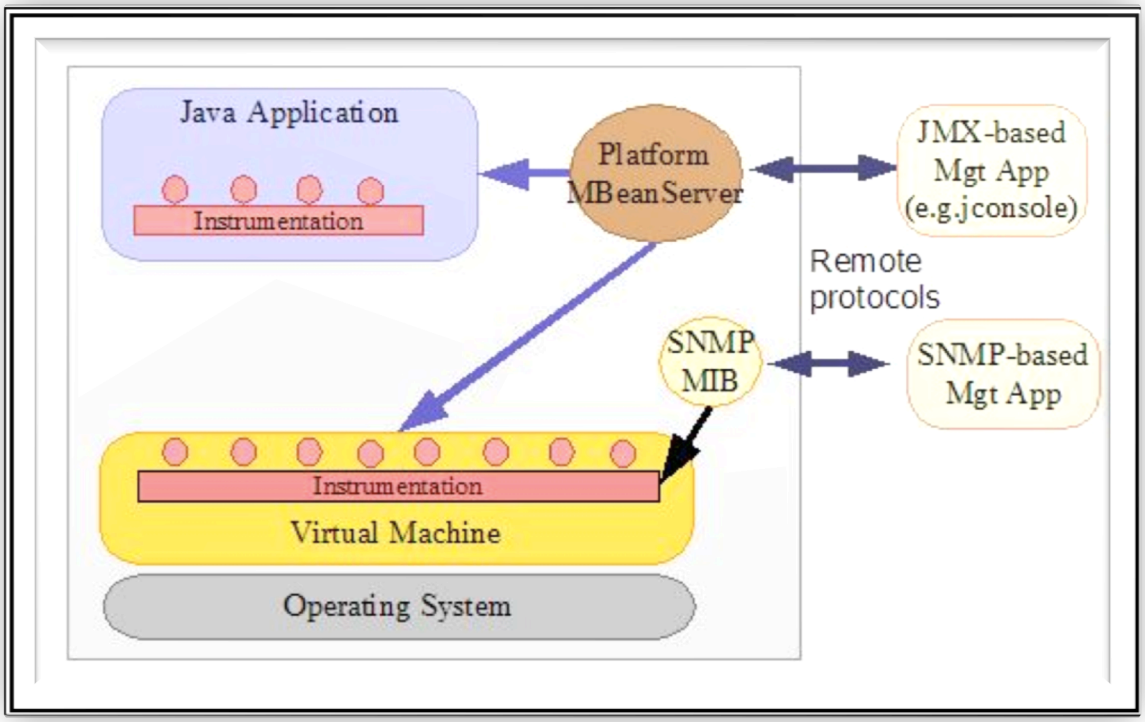


Fig.3.2.0 Arquitectura del monitoreo y administración implementadas en J2SE 5.0

3.1.5 Plataforma MBeans

La plataforma de Java provee un conjunto de MBeans (Managed Beans) que sirven para el monitoreo y la administración de la Máquina Virtual de Java (JVM).

Plataforma MBean	Descripción
Java.lang.management.ClassLoadingMXBean	Carga las clases de sistema de la máquina virtual java.
Java.lang.management.CompilationMXBean	Compilación del sistema de la máquina virtual java.
Java.lang.management.MemoryMXBean	Memoria del sistema de la máquina virtual java.
Java.lang.management.MemoryManagerMXBean	Gestor de la memoria de la máquina virtual java.
Java.lang.management.MemoryPoolMXBean	Espacio de memoria de la máquina virtual java.
Java.lang.management.GarbageCollectorMXBean	Recolector de basura de la máquina virtual java.
Java.lang.management.ThreadMXBean	Sistema de hilos de la máquina virtual java
Java.lang.management.RuntimeMXBean	Tiempo de ejecución de la máquina virtual java.
Java.lang.management.OperatingSystemMXBean	Sistema operativo en el cual la máquina virtual java se ésta ejecutando.

Java.util.logging.LoggingMXBean	Inicio de sesión.

Un MBean es un objeto administrado que sigue un diseño de patrones declarado en la especificación de JMX, este puede representar un dispositivo, una aplicación o cualquier recurso que necesite ser gestionado. La interfaz de gestión de un MXBean comprende el conjunto de los atributos de lectura y escritura y el conjunto de operaciones. MBeans puede también a su vez emitir notificaciones cuando un evento predefinido ocurra.
Cada plataforma MBean tiene un conjunto de atributos y operaciones como el uso de la memoria, el uso del CPU, métricas del recolector de basura, etc. Algunas de estas también emiten a su vez notificaciones.

Jconsole es una herramienta con una interfaz gráfica de usuario (GUI) que se conecta a la JVM la cual empieza a monitorear de forma local al configurar la propiedad com.sun.management.jmxremote.

Para ejecutar JConsole se inicializa con el comando

JDK_HOME/bin/jconsole

Al iniciar la conexión se abre la siguiente caja de diálogo con una lista de conexiones locales corriendo en la JVM

Jconsole se puede conectar de tres diferentes formas:

Local: Jconsole se conecta a la JVM del sistema local el cual es ejecutado con el mismo usuario. Jconsole se conecta a la plataforma MBean usando el conector RMI.

Remota: Jconsole se conecta a la JVM usando el agente JMX con la siguiente URL:

service:jmx:rmi:///jndi/rmi//hostName:portNum/jmxrmi

Donde el hostname y el portNumber son configurados con el agente JMX. Jconsole recibe el usuario y la contraseña para autenticarse usando el conector RMI usando la propiedad “jmx.remote.credentials” para establecer la conexión.

Avanzado: Jconsole se conecta usando el agente JMX(JSR-3) ó JMX(JSR-160) usando la URL anteriormente especificada.

Cuando Jconsole se conecta de forma exitosa, este obtiene información de la JVM y dicha información la despliega en las siguientes pestañas:

Summary Tab: En ésta pestaña se muestra un resumen de la JVM.
Memory Tab: En ésta pestaña se muestra la información del uso de la memoria.
Threads Tab: En ésta pestaña se muestra la información del uso de los hilos.
Classes Tab: En ésta pestaña se muestra la información de las clases.
VM Tab: En ésta pestaña se muestra información de la JVM.
MBeans Tab: En ésta pestaña se muestra la información de todos los MBeans.

La pestaña de MBeans muestra información de todos los MBeans registradas en la JVM. A su vez MBeans permite el acceso a toda la configuración de la plataforma.

La memoria del MBean contiene cuatro atributos:

1. HeapMemoryUsage: Es un atributo de solo lectura que muestra el actual uso de memoria.
2. NonHeapMemoryUsage: Es un atributo de solo lectura que muestra la memoria no usada.
3. ObjectPendingFinalizationCount: Es un atributo de solo lectura que describe el número de objetos pendientes por finalizar.
4. Verbose: Es un atributo de tipo boolean que describe la configuración del recolector de basura el cual a su vez puede ser configurado de forma dinámica.

La memoria de MBean soporta una operación del recolector de basura, los detalles de la interface de MBean se pueden ubicar y observar en la fig. 3.2.2 en java.lang.management.MemoryMXBean.

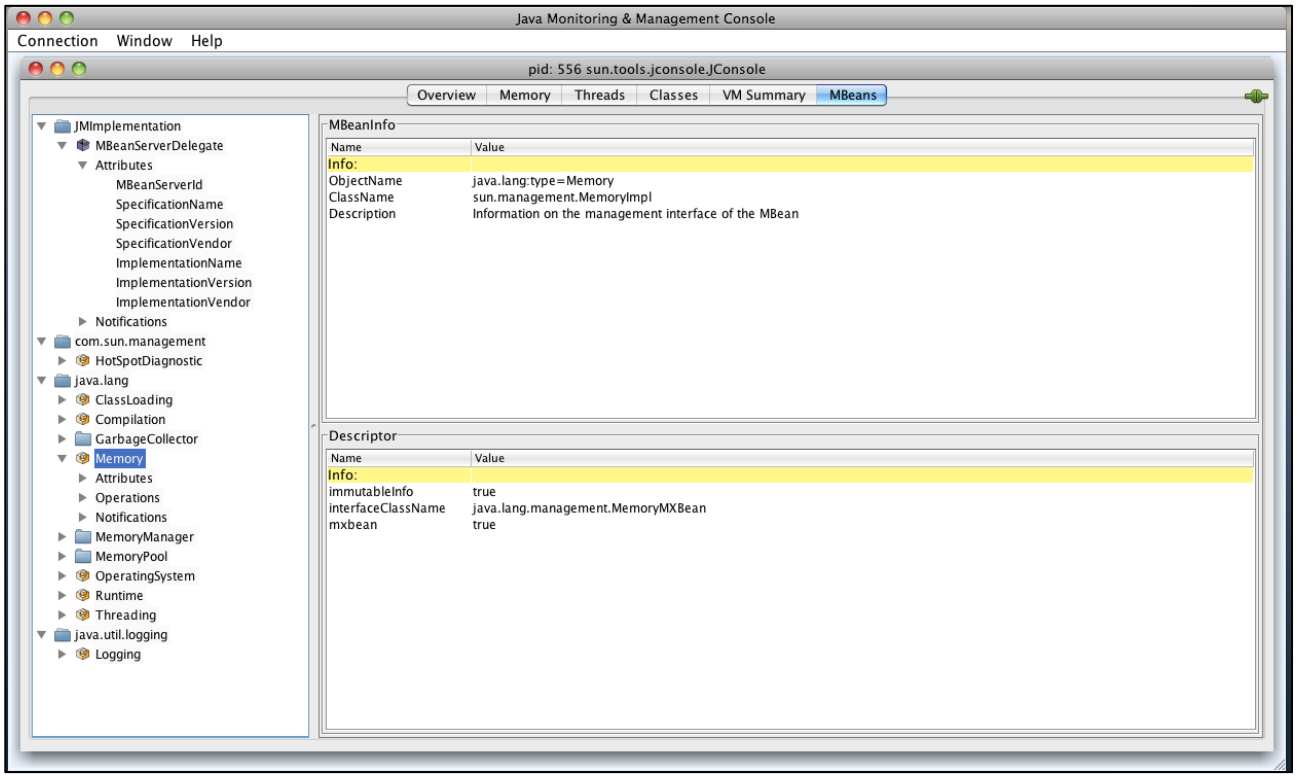


Fig.3.2.2 Pestañas de MBeans

En la figura 3.2.2 se muestra un árbol del lado izquierdo con una lista de todos los MBeans organizado por nombre de objeto. El nombre de un objeto MBean consiste en el nombre del dominio y una lista de propiedades.

Cuando se selecciona un MBean del árbol, sus atributos, sus operaciones y sus notificaciones son desplegados del lado derecho como se muestra en la fig.3.2.3. Aquí se puede configurar el valor de los atributos, de igual forma se pueden llamar operaciones las cuales se pueden observar en la pestaña de operaciones. Además se puede monitorear las notificaciones emitidas por MBean: por default Jconsole no escucha ninguna notificación emitida por MBean hasta que ésta sea publicada. Se puede hacer click en la opción “Subscribe” para publicarla o en la opción “Unsubscribe” para no publicarla.

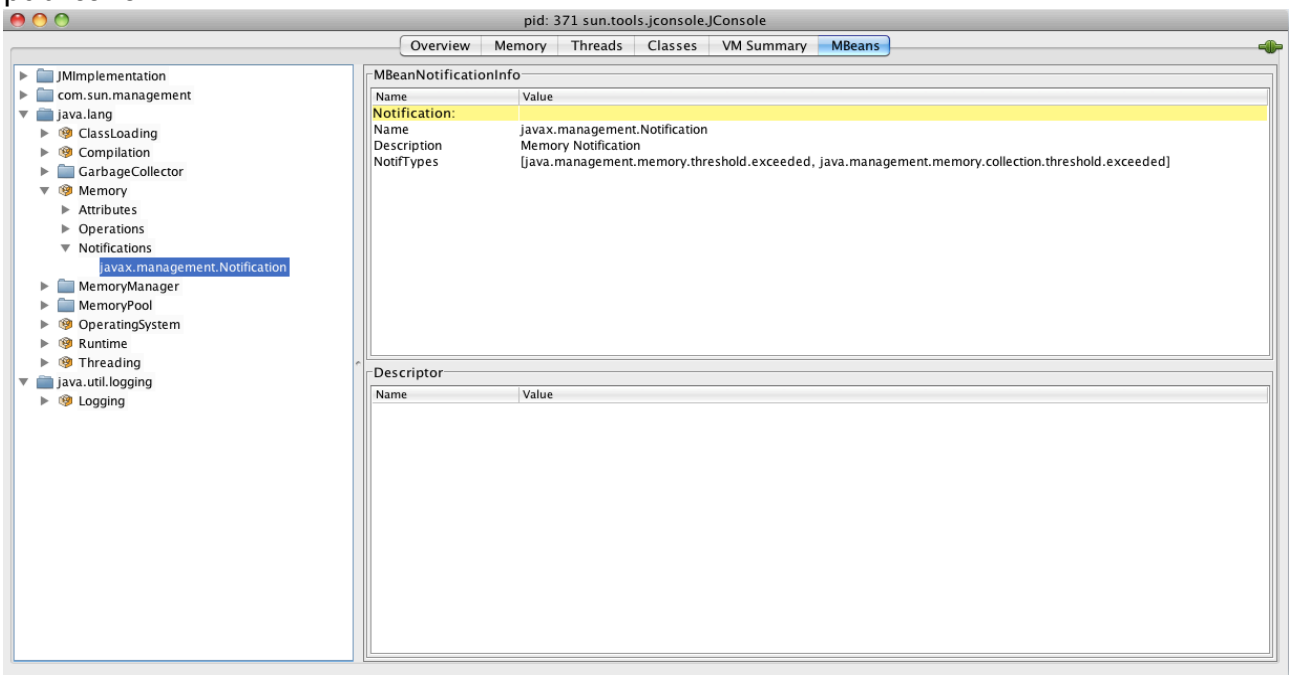


Fig.3.2.3 Notificaciones de MBeans

3.1.6 Detección del consumo de memoria.

La pestaña de memoria provee información acerca del consumo de memoria, espacio de memoria, métricas del recolector de basura como se puede observar en la fig.3.2.4

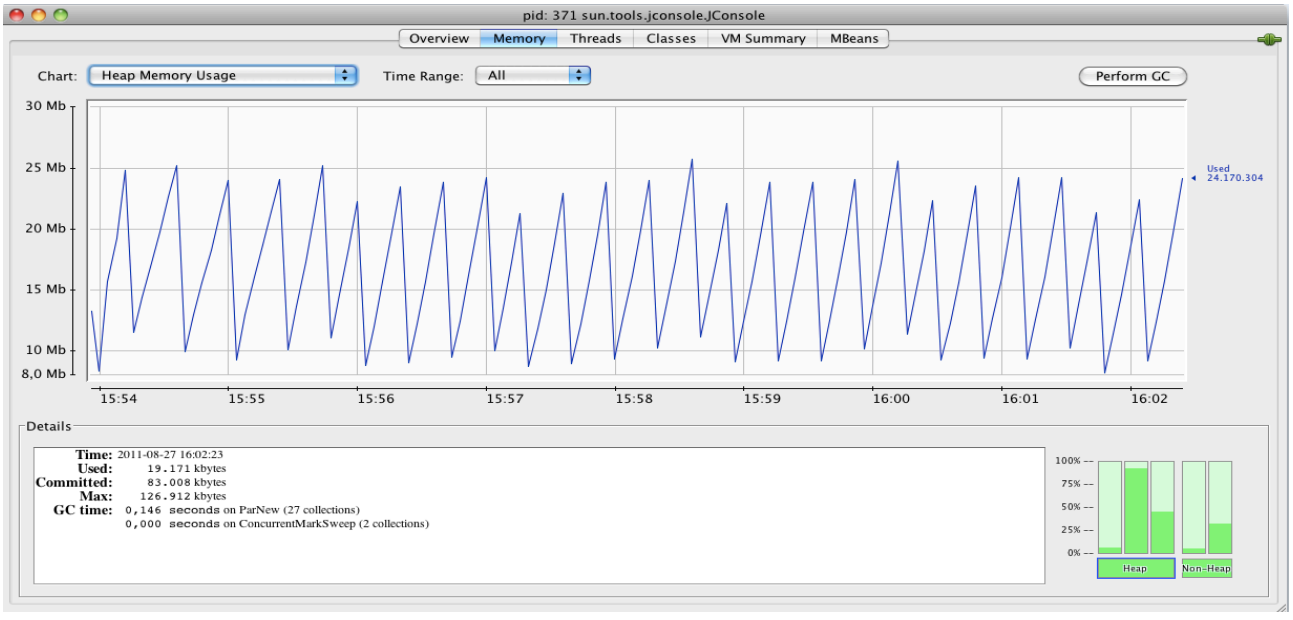


fig.3.2.4 Pestaña de memoria

En la siguiente lista se muestra los diferentes bloques de la máquina virtual.

- Eden Space: Aquí es donde se crean los objetos inicialmente.
- Survivor Space: Es como un tipo de espacio del cual se pasa de la generación joven a la generación vieja, suele estar compuesto a su vez de dos partes. Se puede especificar en proporción entre el espacio de la generación joven y la generación vieja con el comando de la máquina virtual.

  -xx:new ratio

- Ternured Generation: Es un espacio que contiene objetos que han existido durante un periodo de tiempo en el Survivor Space.
- Permanent Generation: Contiene todos reflejados de la máquina virtual java tales como clases, métodos y objetos. Con la JVM corriendo ésta generación ésta dividida en área de solo escritura y solo lectura.
- Code Cache: Contiene memoria usada a partir de la compilación y el almacenamiento del código nativo.

Los detalles de las métricas se muestran a continuación.

- Used: Es la cantidad de memoria actualmente usada. Ésta incluye la memoria ocupada por los objetos.
- Commited: Es la cantidad de memoria reservada para ser usada por la JVM. La JVM puede liberar la memoria del sistema y comprometida, ésta puede ser menos memoria de la que inicialmente ésta asignada al arranque. La memoria asignada será siempre mayor o igual que la memoria usada.
- Max: La cantidad máxima de memoria puede ser gestionada, su valor puede cambiar o puede no estar definido. El alojamiento de la memoria asignada puede fallar si la memoria de la JVM empieza a incrementarse, el uso de la memoria puede ser mayor que la memoria reservada si la cantidad de memoria es menor o igual que la memoria máxima (por ejemplo cuando el sistema ésta corriendo lento).
- GC Time: Es el tiempo usado por el recolector de basura y el número total de llamadas. Esto puede tener múltiples filas y cada una de ellas representa el algoritmo usado por el recolector de basura en la JVM.

3.1.7 Memory Thresholds

Un grupo de memoria pueden tener dos tipos de memory threshold: Usage threshold y Collection Usage Threshold.

Usage Threshold es un atributo manejable permite monitorear el uso de memoria con bajo costo operativo. Si se establece un valor positivo permite el monitoreo del comportamiento de memoria. Si se establece el valor cero se desactiva el uso como se observa en la fig.3.2.5

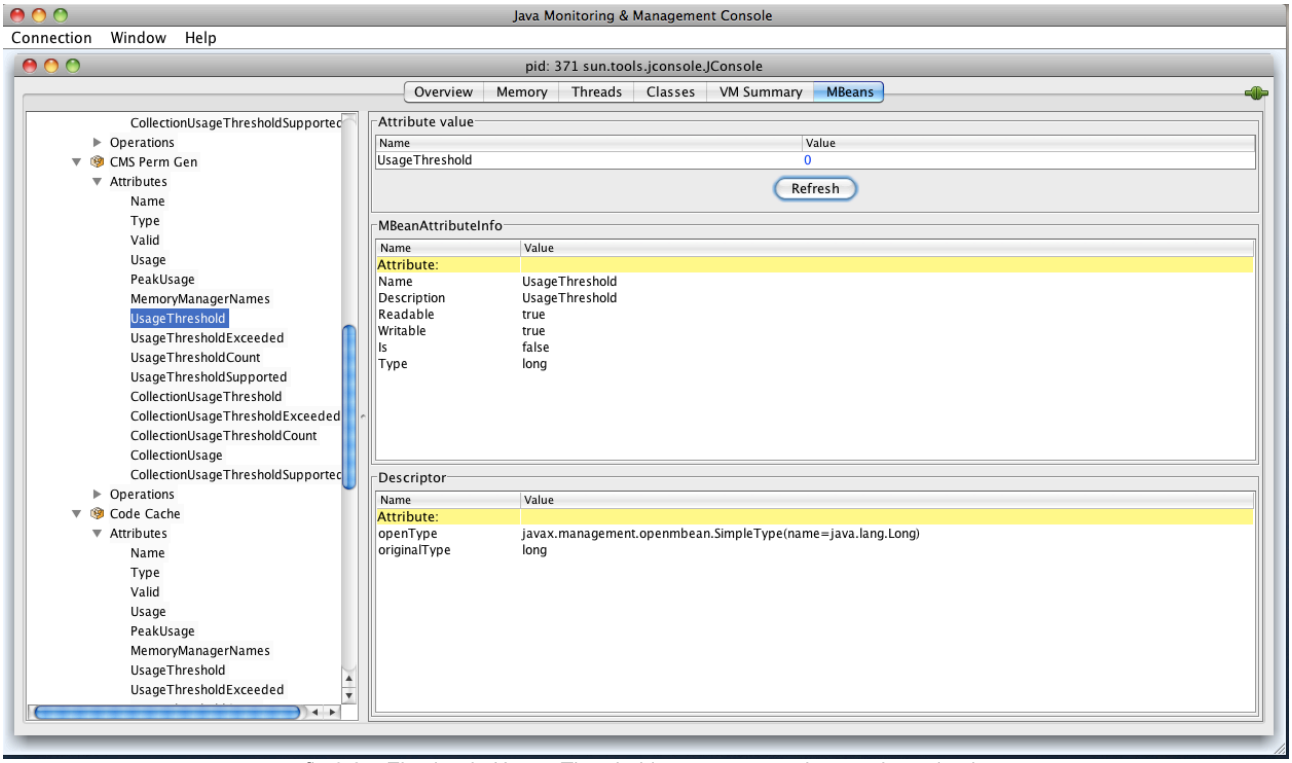


fig.3.2.5 El valor de Usage Threshold es cero por ende esta desactivado

Collection Usage Threshold

Collection Usage Threshold es un atributo manejable de algunos grupos de memoria. Después de que la JVM ha llevado a cabo la recolección de basura en un banco de memoria, cabe la posibilidad de que algunos objetos aún estén en uso. Si al atributo se le establece un valor positivo se habilitará el monitoreo. Si se establece el valor cero se desactiva el uso como se observa en la fig.3.2.6

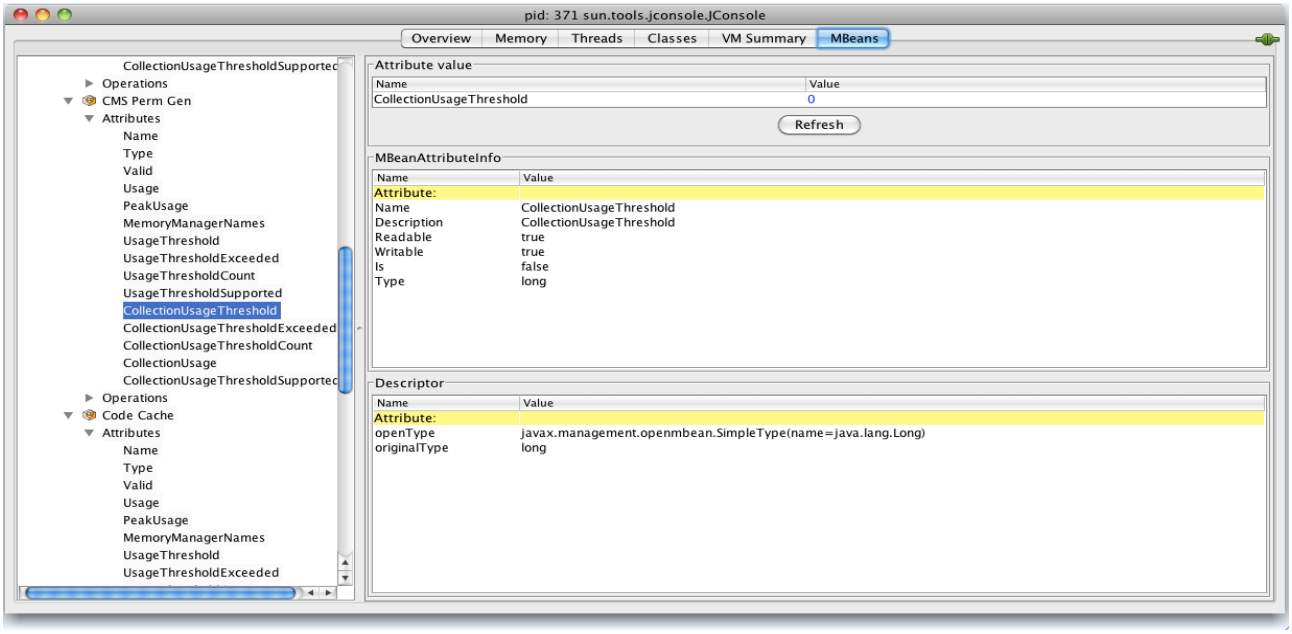


fig.3.2.6 El valor de Collection Usage Threshold es cero por ende esta desactivado

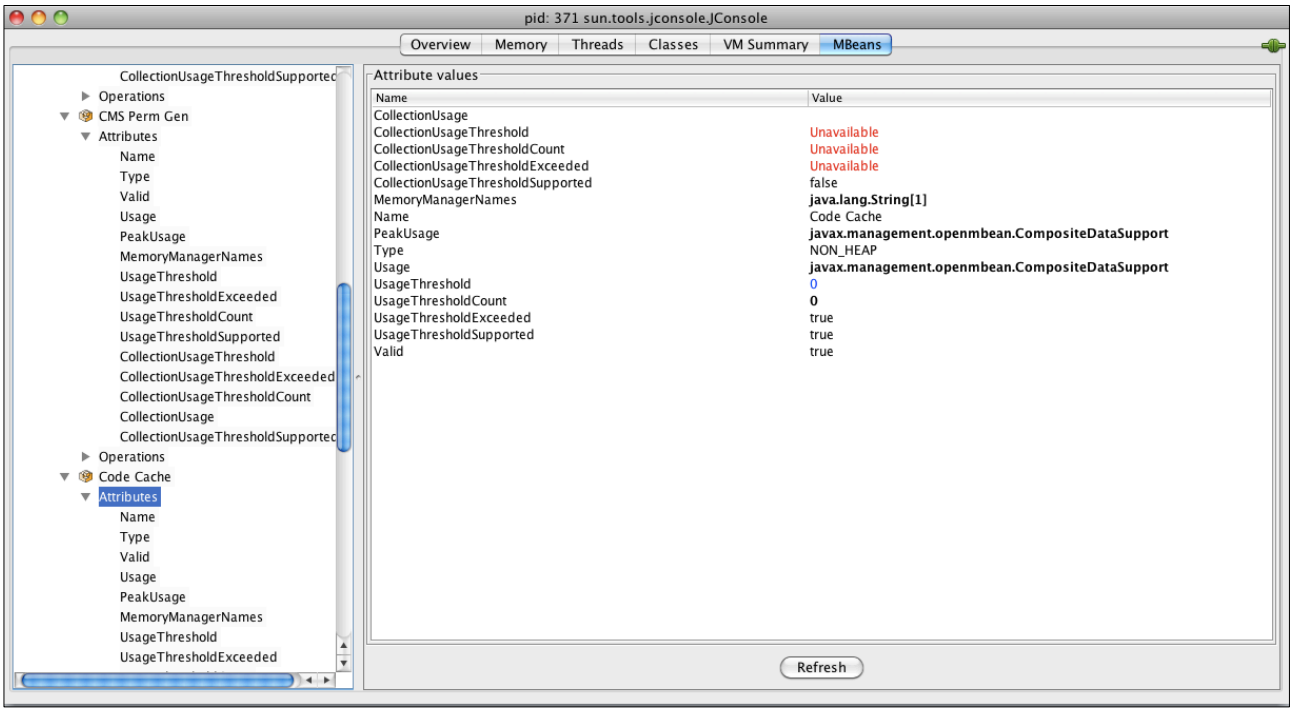


Fig.3.2.7 Configuración de usage threshold y de collection usage threshold.

3.1.8 Habilitación y deshabilitación del parámetro VM Verbose.

Como se menciono anteriormente, la memoria del sistema MBean ésta definido por un atributo booleano llamado GC verbose el cual permite encender o apagar el rastreo dinámicamente. El GC Verbose despliega la locación definida cuando la JVM inicia como se muestra en la fig.3.2.8

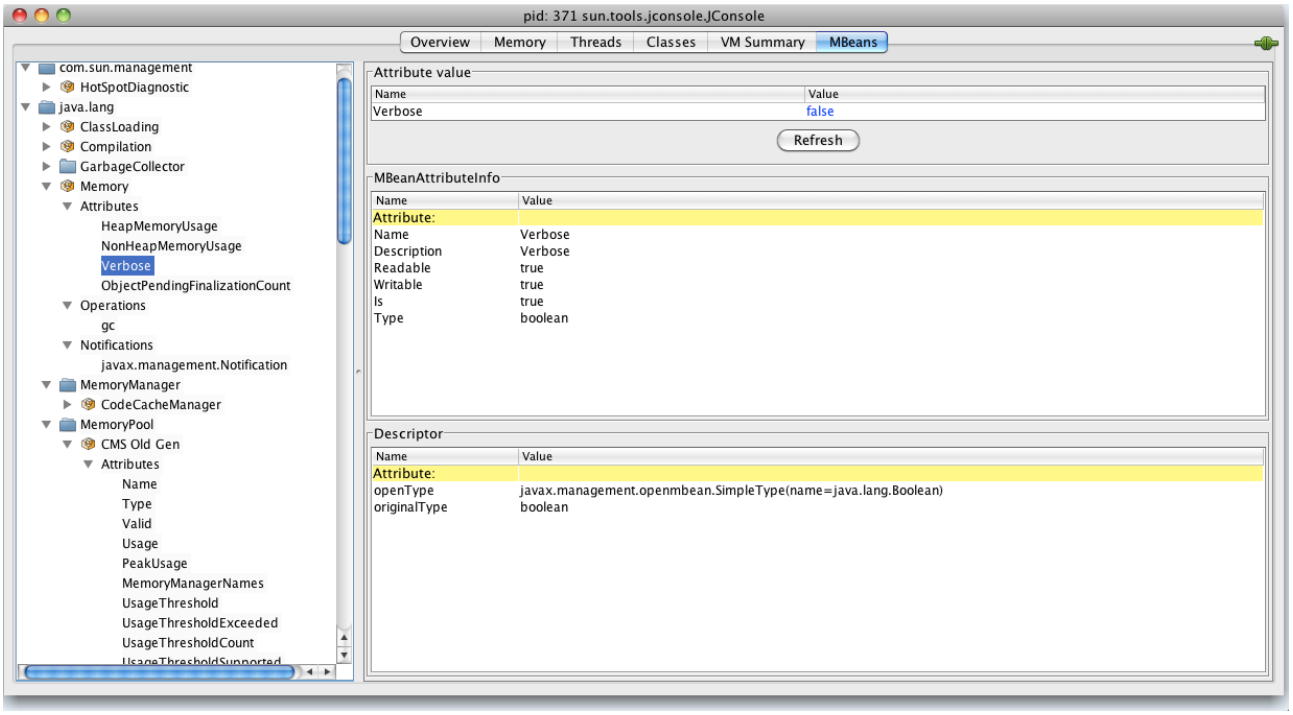


Fig.3.2.8 Configuración de Verbose GC.

3.1.9 Detección de cuellos de botella.

La pestaña de threads provee información acerca de los hilos que están corriendo en la aplicación como se muestra en la fig.3.2.9

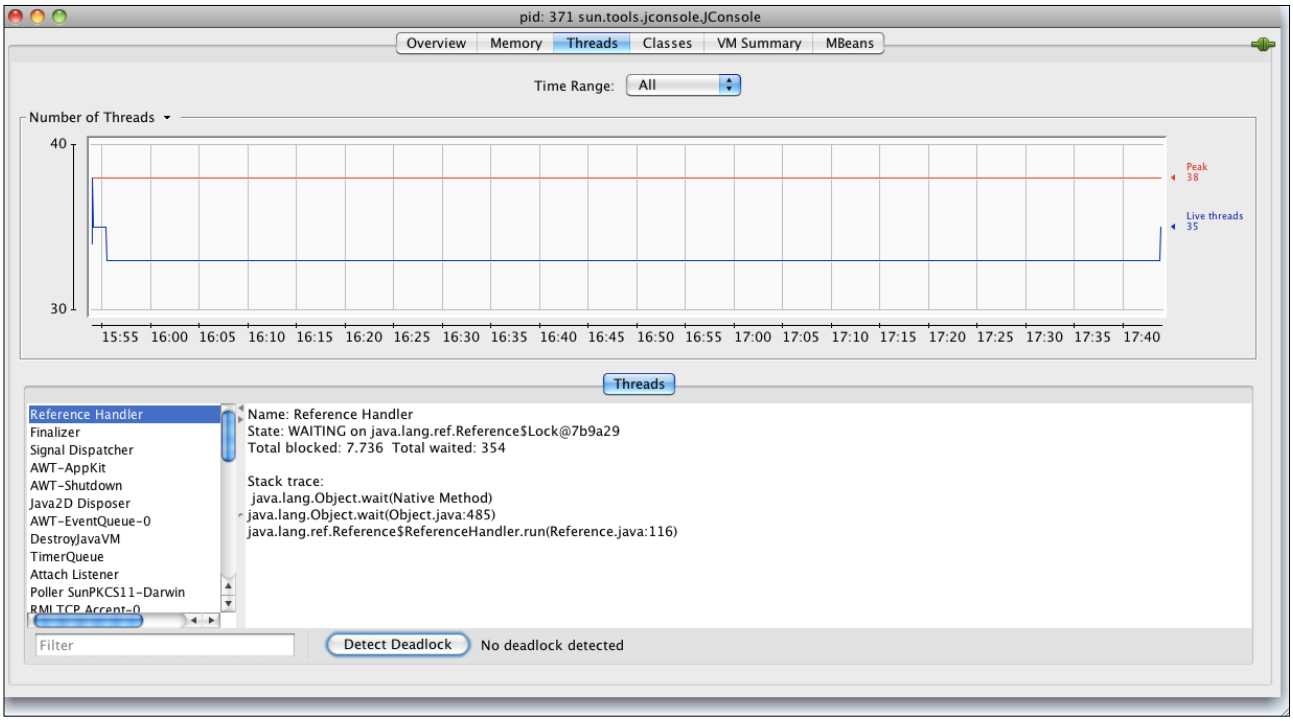


fig.3.2.9 Pestaña de Threads.

En la fig.3.2.9 se muestra una lista de los hilos activos. Si se teclea una cadena de caracteres en el campo Filter la herramienta hará una búsqueda y filtrará todos los hilos que contiene la cadena de caracteres que se ingresó previamente.

Mbeans provee otro tipo de información útil la cual no ésta incluida en la pestaña Threads como son:

findMonitorDeadlockedThreads: Detecta cualquier hilo que tenga un cuello de botella. Ésta operación regresa un arreglo de los IDs de los hilos que tengan cuello de botella.

getThreadInfo: Regresa la información del hilo, incluye el nombre, rastreo de la pila y si el monitor ésta bloqueado muestra la información del hilo que lo bloqueo y conexiones a los hilos estáticas.

getThreadCpuTime: Regresa el tiempo que utiliza la CPU dado por el hilo.

Para acceder estas diferentes características, hay que seleccionar la pestaña de Mbeans y seleccionar Threading MBean dentro del árbol Mbeans. Esto muestra todos los atributos y operaciones para acceder a la información de la JVM.

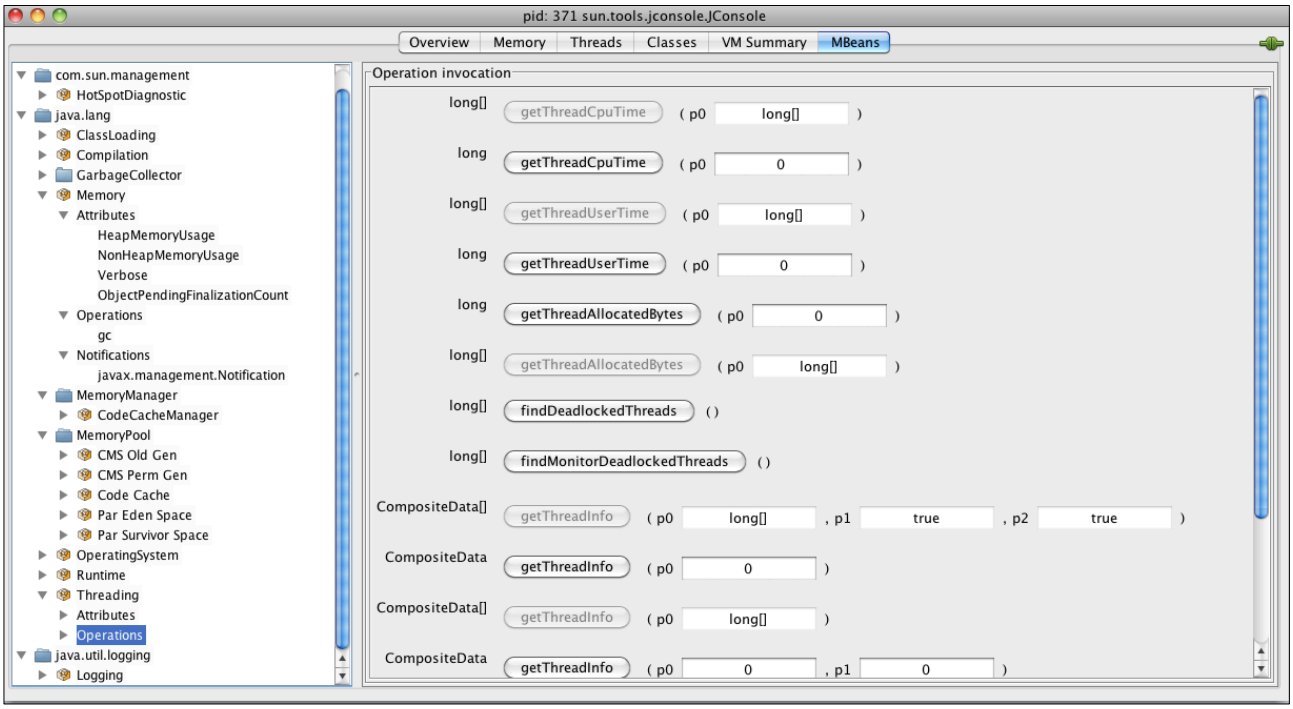


fig.3.3.0 Pestaña de Mbeans Threading

Un cuello de botella, se puede observar mediante findMonitorDeadlockedThreads como se observa en la fig. 3.3.1

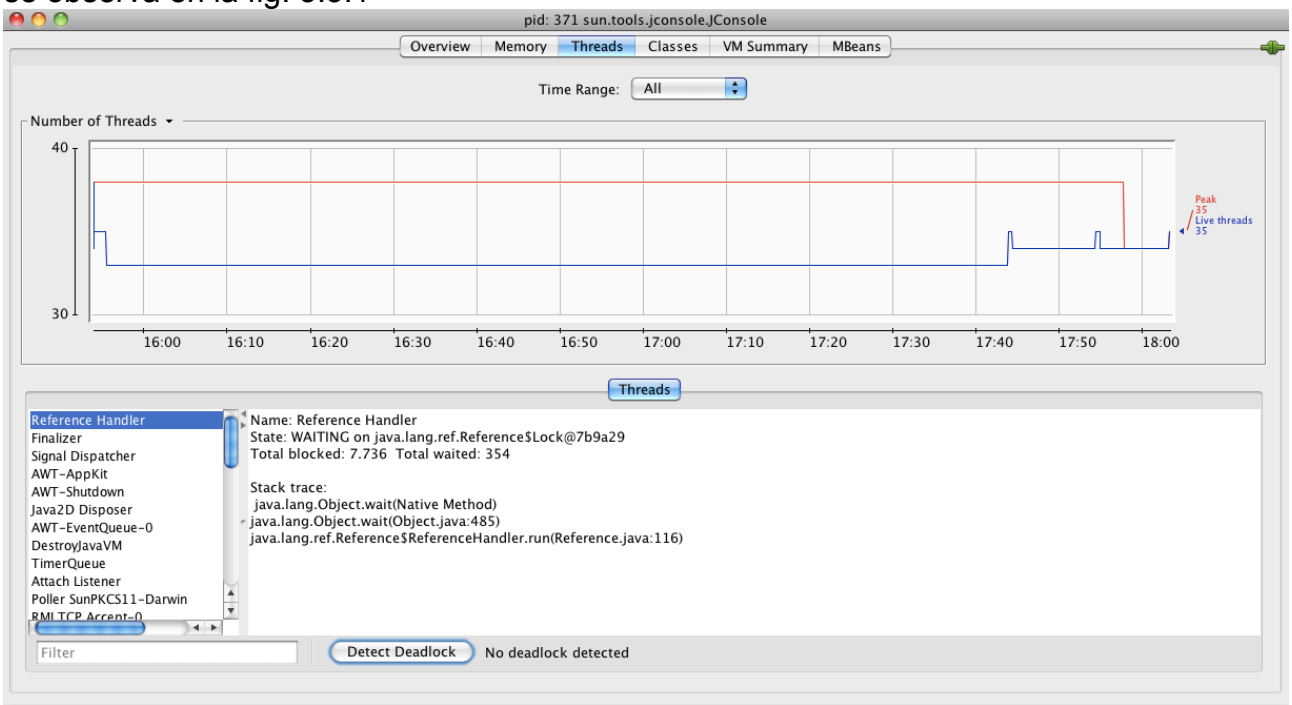


fig.3.3.1 Busca los hilos con cuello de botella.

Una vez que se haga click en la pestaña findMonitorDeadlockedThreads. La operación regresa un valor mediante una ventana que funciona como pop up como en la fig.3.3.2 JConsole se conecta a la aplicación y hace una prueba y ésta muestra un árbol de hilos que se encuentran en un cuello de botella. Para mas información acerca de los cuellos de

botella puedes usar el método getThreadInfo operation. El Threading MBean soporta el método getThreadInfo operation de cuatro diferentes formas, las cuales pueden obtener la siguiente información:

- Dado el ID de un hilo con el trace de la pila y el número máximo de frames.
- Dado el arreglo del ID de un hilo con el trace de la pila y el número máximo de frames.
- Dado el ID de un hilo sin el trace de la pila.
- Dado el arreglo del ID de un hilo sin el trace de la pila.

Para un deadlocked, el administrador a su vez también estaría interesado en el stack trace. Para esto basta con introducir el ID del hilo correspondiente al deadlocked cuyo primer parámetro del método getThreadInfo es el ID y como segundo parámetro se define el número de estructuras que se desea obtener.

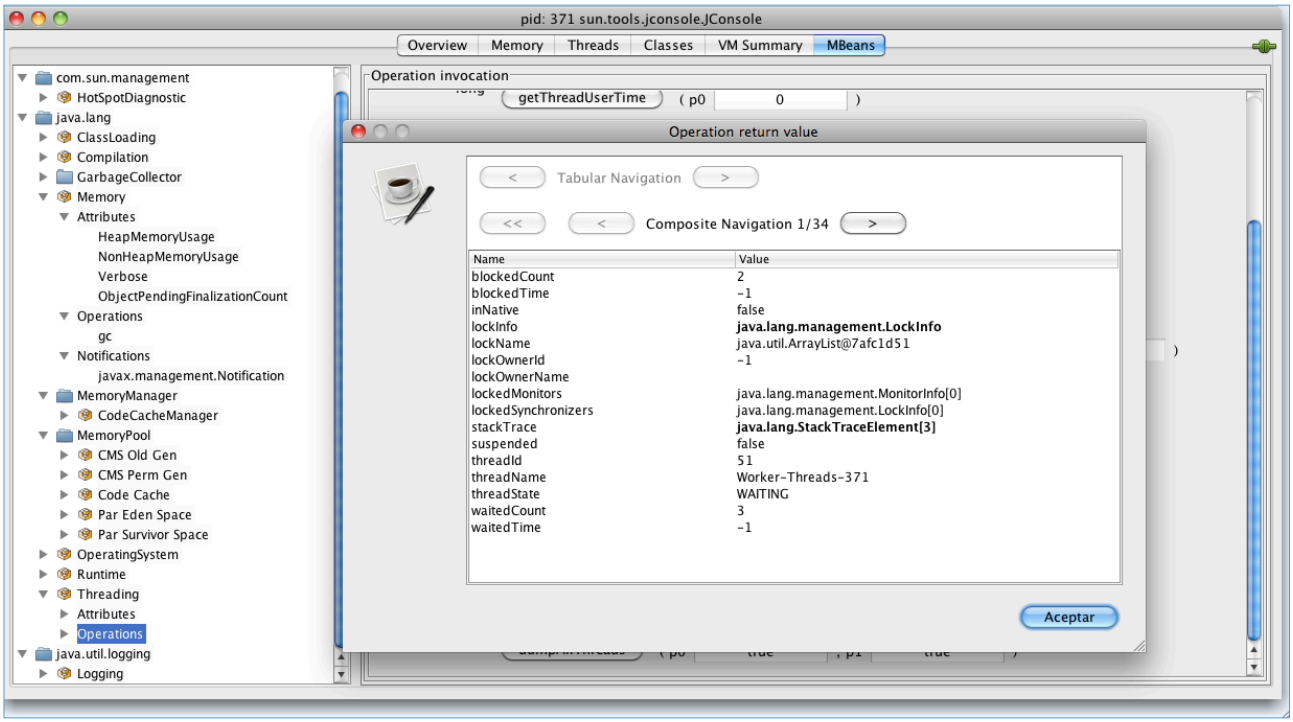


Fig. 3.3.2 ThreadInfo para un PID = 371

3.2.0 Control de nivel de registro.

El Registro de MBean LoggerNames define un atributo que describe la lista de nombres. Para encontrar la lista en la aplicación, seleccione la pestaña MBean como se muestra en la fig.3.3.3

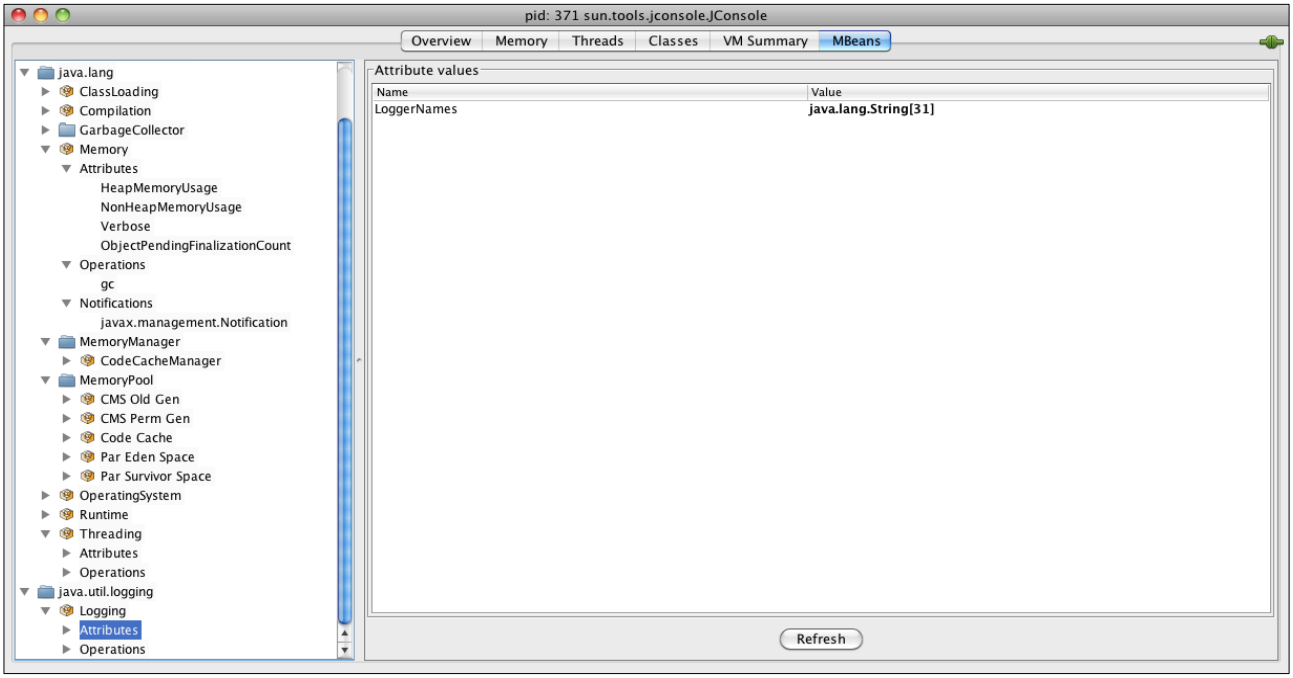


Fig 3.3.3 Lista de loggernames

El MBean es compatible con tres operaciones:

- getParentLoggerName. Devuelve el nombre de log padre.
- getLoggerLevel. Devuelve el nivel de log.
- setLoggerLevel. Establece el nivel de log .

Las tres operaciones tienen un nombre. Para cambiar el nivel de un registrador, hay que escribir el nombre del registrador en el primer parámetro y el nombre de la instancia en el segundo parámetro de la operación setLoggerLevel y así ajustamos el nivel de registro como se ve en la fig.3.3.4

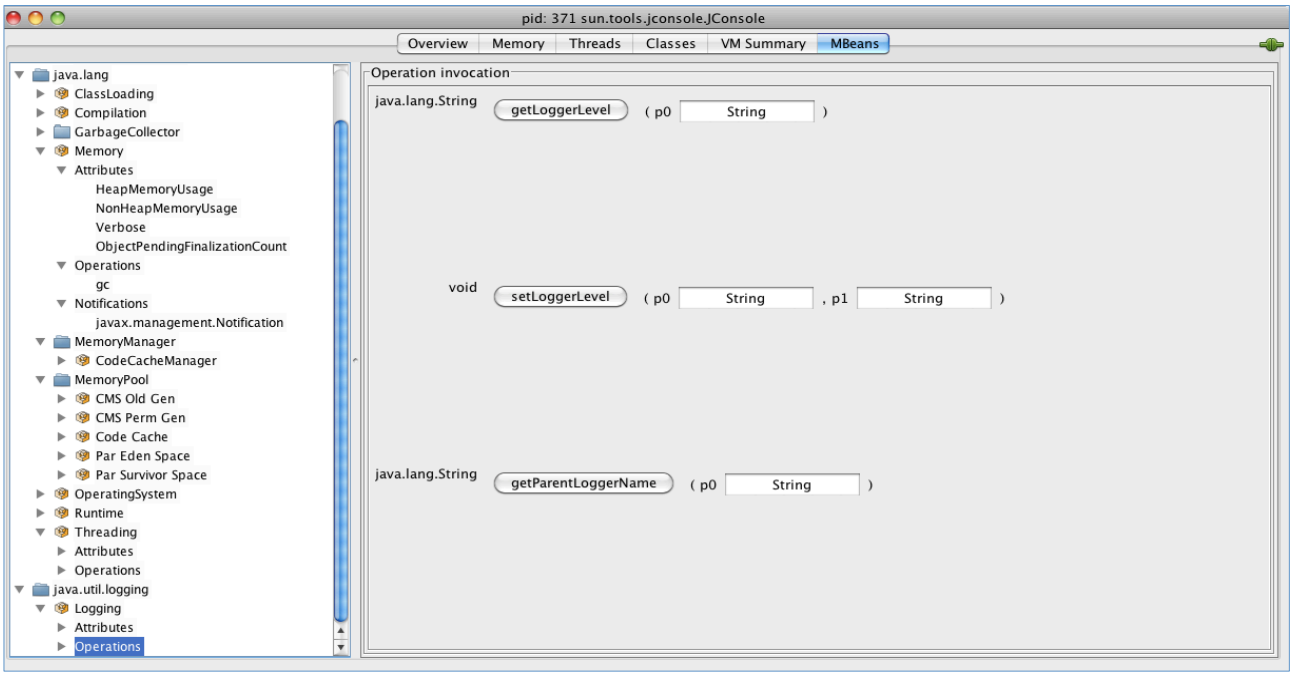


fig 3.3.4 Ajuste del nivel de registro

3.2.1 Acceso a los Recursos Extensión OS de Sun Plataforma

El JDK 5.0 integra en el sistema operativo MBean para obtener información de recursos usados tales como:

- El tiempo de procesamiento de CPU.
- La cantidad de memoria física total y libre.
- La cantidad de memoria virtual comprometida (es decir, la cantidad de memoria virtual garantizada disponible para el proceso en ejecución).
- El número de descripciones de archivo abierto (sólo UNIX).

En la fig. 3.3.4 se ha seleccionado la pestaña MBeans, y en ella se ven todos los atributos y operaciones, incluyendo la extensión de la plataforma. En dicha pestaña se puede monitorear los cambios de un atributo numérico por ejemplo, el tiempo de procesamiento del CPU.

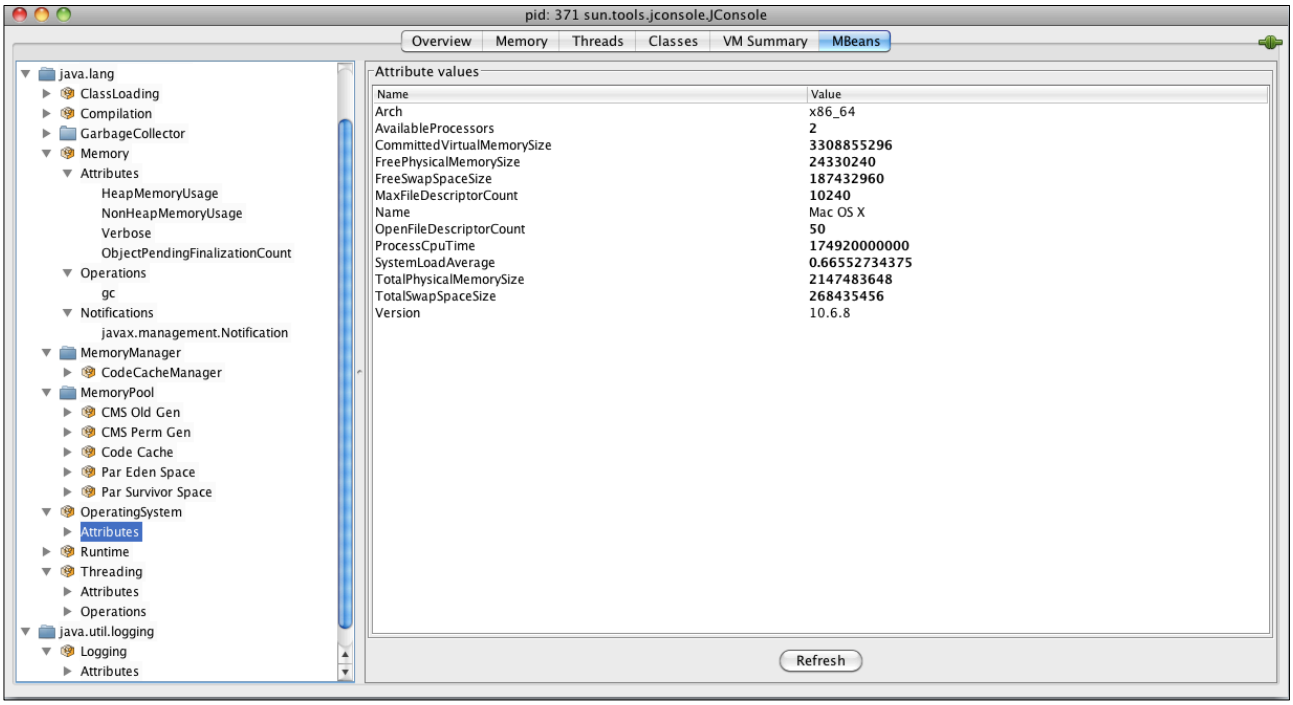


Fig. 3.3.4 MBeans.

Además, en la pestaña VM y en la pestaña Resumen nos proporcionan información sobre los recursos del sistema operativo.

3.2.2 Gestión de aplicaciones MBeans

En la fig. 3.3.5 si el atributo CacheSize es cambiado, el MBean enviará una notificación. Se pueden utilizar MBeans para gestionar la aplicación.

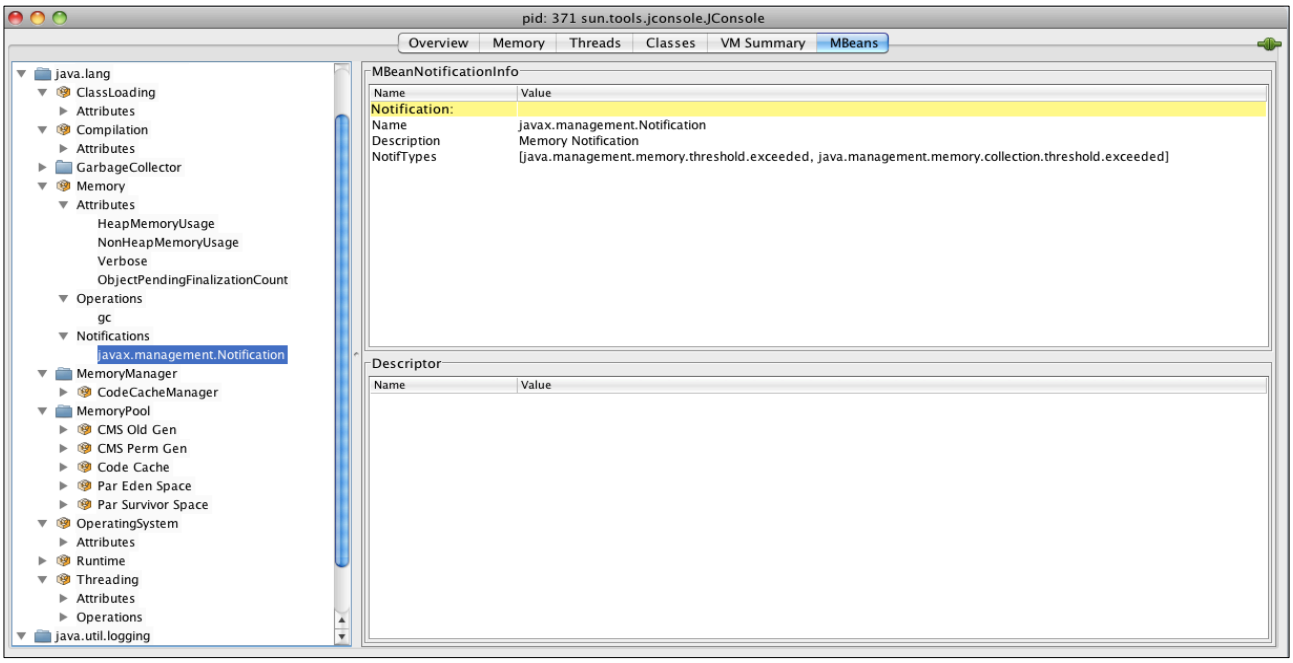


fig. 3.3.5 Notificaciones

Se puede utilizar JConsole para conectarse y observar el funcionamiento de la máquina virtual de Java, monitorear y la actividad de los hilos. Se puede obtener información sobre las clases, información sobre la máquina virtual Java y el sistema operativo.

Comparativo

	VisualVM	Jconsole
Características	1. Conexión local y remota 2. Despligue de la informacion de la aplicación en tiempo real 3. Monitoreo de consumo de memoria en tiempo real 4. Monitoreo de hilos 5. Permite el análisis de la memoria en tiempo real 6. Análisis de hilos 7. Análisis de picos de memoria 8. Bajo consumo sobre la aplicación	1. Análisis de las clases 2. Análisis de la JVM 3. Análisis de la memoria de la JVM 4. Análisis del pool de la JVM 5. Análisis de la GC 6. Threading system para la JVM 7. Runtime system para la JVM 8. Sistema Operativo en el cual el JVM ésta corriendo 9.- Bajo consumo sobre la aplicación
Licencia	Open Source	Open Source
Precio	NA	NA

CAPITULO IV

Estrategia de Solución y Resultados

4.1.0 Estrategia de Solución

La estrategia para la solución al problema de performance de CMM fué la siguiente: Programar una clase java para obtener en número de procesadores bajo los cuales corre la aplicación

Estrategia
Configuración de Recolector de Basura.

```
public class GetJVMInfo
{
  …

    Runtime.getRuntime().totalMemory();
    Runtime.getRuntime().freeMemory();
    Runtime.getRuntime().maxMemory();
    Runtime.getRuntime().availableProcessors();

    System.out.println("Heap Size = "+ heapSize);
    System.out.println("Free Memory = "+ freeMem);
    System.out.println("Max Memory for JVM = "+ maxMem);
    System.out.println("Available Processors =" + processors);
…

  }
}
```

El resultado fué:

Heap Size = 1032847360
Free Memory = 1027436688
Max Memory for JVM = 1032847360
Available Processors =16

Como podemos observar la aplicación corre bajo 16 procesadores, 8 de ellos funcionan como respaldo en caso de que la aplicación tenga alguna falla, procesadores reales son 8. Al analizar la configuración de la máquina virtual se encontró:

JAVA_OPTS="$JAVA_OPTS -d64 -server -Xloggc:/tmp/heap_prod.log -XX:+PrintGCDetails
-XX:+UseCMSInitiatingOccupancyOnly -XX:+PrintGCTimeStamps -XX:+PrintClassHistogram
-XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:+CMSParallelRemarkEnabled -Xss1536k
-Djava.library.path=$JAVA_LIBRARY_PATH -Xmx12g -Xms12g -XX:PermSize=256m -**XX:SurvivorRatio=8**
-XX:TargetSurvivorRatio=90 -**XX:MaxTenuringThreshold=8** -**XX:ParallelGCThreads=4**
-XX:CMSInitiatingOccupancyFraction=40 -XX:+UseCMSInitiatingOccupancyOnly
-XX:+DisableExplicitGC -XX:MaxNewSize=4g -XX:NewSize=4g -Dfile.encoding=ISO-8859-1
-Duser.timezone=GMT -Dcom.trema.commons.config.base=/….
Dopenorb.char.encoding=UTF-8 -Dopenorb.wchar.encoding=UTF-16 -Dcatalina.base=$CATALINA_BASE"

Según la documentación de la JVM la formula para configurar el Garbage Collector es:

ParallelGCThreads = (#cpus <= 8) ? #cpus : 3 + ((#cpus * 5) / 8)

Ejemplo

Si tenemos

#cpus=4, ParallelGCThreads=4

#cpus=8, ParallelGCThreads=8

#cpus=16, ParallelGCThreads=13

Por ende si tenemos que la aplicación corre bajo 8 procesadores la configuración del Garbage Collector en paralelo tendría que ser 8.

XX:ParallelGCThreads=4

Para ello se sugerieron los siguientes parámetros.

	Opciones de Desempeño	
Valor Actual	**Valor sugerido para el ambiente de produccioón**	**Descripción**
	-XX:MaxPermSize=192m	MaxPermSize especifica el tamaño máximo para la cantidad de memoria utilizada cuando los objetos, clases y métodos son cargados. El valor por defecto es de 64 MB Si la aplicación necesita mas memoria para cargar sus objetos entonces se

		lanza un error java.lang.OutOfMemoryError que se observará en los logs por otro lado si se reduce este valor vamos a aprovechar el tamaño de pila.
-XX:SurvivorRatio=8	-XX:SurvivorRatio=16	El índice de supervivencia es la proporción entre el espacio de objetos nuevos (eden) y el espacio de supervivientes de la sección de objetos jóvenes del almacenamiento dinámico. Al aumentar este valor se optimiza la JVM para las aplicaciones en las que se crean muchos objetos y se conservan pocos objetos. El valor por defecto es 32, el valor recomendado es 16.
-XX:MaxTenuringThreshold=8	-XX:MaxTenuringThreshold=31	Este parámetro se refiere al número máximo de veces que se puede copiar un objeto vivo entre los espacios de supervivencia antes de ser trasladado al espacio de titular. El valor por recomendado es 31
	-Dsun.rmi.dgc.server.gcInterval=600000 -Dsun.rmi.dgc.client.gcInterval=600000	Con estos valores podemos establecer el intervalo de tiempo en milisegundos para la ejecución del recolector de basura (10 minutos) en la máquina hotspot que hemos elegido.
	Debugging Options	
-XX:ParallelGCThreads=4	-XX:ParallelGCThreads=8	Establece el número de hilos del recolector de basura configurados en paralelo. Este valor se obtiene con la fórmula: -XX:ParallelGCThreads=(#CPUs<8? #CPUs : 3 + ((5*#CPUs)/8))

		donde #CPUs=8 (#CPUs usados en CMM)
	Behavioral Options	
-XX:CMSInitiatingOccupancyFraction=40	-XX:CMSInitiatingOccupancyFraction=68	Este parámetro establece el porcentaje máximo de la vieja generación donde se activa el GC. El valor recomendado para la version 1.5 de java es de 68

4.1.1 Análisis de logs.

Hibernate. Los DAO's son un patrón de diseño del core de j2ee y son considerados una buena práctica de programación se usan para aislar la aplicación de la persistencia la cual podría ser JDBC, EJB o Hibernate. Esto entrega flexibilidad pero su desventaja es que al incrementar la capa de persistencia incrementa la cantidad de código ejecutado durante el tiempo de ejecución por ende no se recomienda que las aplicaciones de rendimiento critico utilicen DAO's.

Después del análisis de logs y se encontró:

FK_IDENT=iclqa2, 2011-01-12 12:49:43,623, INFO [Thread-14[com.trema.acm.erp.daemon.DocumentReaderDaemon]] (HibernateLockDAO.java:58) - Ping ACMLock:DocumentReader Version:2866115Time:Wed Jan 12 08:06:01 GMT 2011
FK_IDENT=iclqa2, 2011-01-12 12:49:43,631, DEBUG [Thread-14[com.trema.acm.erp.daemon.DocumentReaderDaemon]] (HibernateLockDAO.java:52) - LastPing=~CurrentTime => no ping (ignore time:2500

La recomendaciones por el equipo de Middleware son:

1. Analizar y verificar los querys.
2. Una vez hecha la consulta cerrar el pool de conexiones.
3. Analizar la arquitectura sobre todo en la capa de persistencia.

Resultados.

Se aplico la misma lógica para hacer el análisis de los parámetros de la maquina virtual en los ambientes de desarrollo y de calidad siendo estos satisfactorios, posteriormente se aplicaron los cambios en el ambiente de producción y hasta el momento se ha observado notable mejoría en el desempeño de dicha aplicación. El equipo indú de desarrollo por su parte corrigieron los querys, procuraron cerrar sus conexiones después de las consultas y mejoraron el desarrollo en la capa de la persistencia.

CONCLUSIONES

Java está pensado para ser independiente de la arquitectura y la manera que tiene Java para conseguirlo es a través de la emulación de una máquina software sobre la que funcionan los programas compilados con Java. El JVM se encarga de interpretar el archivo .class que genero el compilador.
Un recolector de basura es un mecanismo implícito de gestión de memoria implementado en el lenguaje de programación java.
Cualquier programa hace uso de una cierta cantidad de memoria de trabajo puesta a su disposición por el SO. Ésta memoria tiene que ser gestionada por el propio programa para:

1. Reservar espacios de memoria para su uso.
2. Liberar espacios de memoria previamente reservados.
3. Compactar espacios de memoria libres y consecutivos entre sí.
4. Llevar cuenta de qué espacios están libres y cuáles no.

Generalmente, el programador dispone de una biblioteca de código que se encarga de estas tareas. No obstante, el propio programador es responsable de utilizar adecuadamente ésta biblioteca.
Esto tiene la ventaja de que se hace un uso eficiente de la memoria, es decir, los espacios de memoria quedan libres cuando ya no son necesarios. Sin embargo, este mecanismo explícito de gestión de memoria es propenso a errores. Por ejemplo, un programador puede olvidar liberar la memoria de manera que, tarde o temprano, no quede memoria disponible, abortando la ejecución del programa.
Como alternativa es necesaria una gestión implícita de memoria, donde el programador no es consciente de la reserva y liberación de memoria.
Cuando un lenguaje dispone de recolección de basura, el programador no tiene que invocar a una subrutina para liberar memoria. La reserva de memoria también es más o menos automática sin la intervención del programador. Por ejemplo:

En los lenguajes orientados a objetos (JAVA): se reserva memoria cada vez que el programador crea un objeto, pero éste no tiene que saber cuánta memoria se reserva ni cómo se hace esto.
En los lenguajes declarativos: cada vez que se construye una expresión se reserva memoria (de una manera inteligente), pero el programador no es consciente de ello.

Cuando se compila el programa, automáticamente se incluye en éste una subrutina correspondiente al recolector de basura. Ésta subrutina también es invocada periódicamente sin la intervención del programador.
El recolector de basura es informado de todas las reservas de memoria que se producen en el programa. Además, el compilador colabora para que sea posible llevar una cuenta de todas las referencias que existen a un determinado espacio de memoria reservado. Es decir si la aplicación o el programa está corriendo y ocupa el 95% de la CPU el recolector de basura reclamara la memoria de todas las instancias de memoria no utilizadas al igual lo hace si el programa ocupa el 50% o el 75% de la CPU.

Existen varias formas de configurar el GC y dado que nuestra aplicación corre sobre 8 procesadores se aplica lo siguiente:

Ejemplo

Cuando #cpus=4, ParallelGCThreads=4
Cuando #cpus=8, ParallelGCThreads=8
Si tenemos mas de 8 procesadores se aplica la siguiente fórmula para determinar la configuración del GC

ParallelGCThreads = (#cpus <= 8) ? #cpus : 3 + ((#cpus * 5) / 8)

Cuando se invoca el recolector de basura, recorre la lista de espacios reservados observando el contador de referencias de cada espacio. Si un contador ha llegado a cero significa que ese espacio de memoria ya no se usa y, por tanto, puede ser liberado. Naturalmente, este proceso consume un cierto tiempo en el que no se hace nada verdaderamente útil para el propósito del programa. Por tanto, no puede ser invocado con demasiada frecuencia.
Así mismo, tener objetos en memoria sin usar, aunque no molesten a la ejecución del programa, ralentizan la ejecución del Recolector de Basura, porque tendrá que procesarlos una y otra vez en todas sus pasadas. En algún momento puede parecer tentador forzar una ejecución del Recolector de Basura llamando a System.gc(). Sin embargo, esto lo único que hará será forzar de forma asíncrona, rompiendo completamente toda la heurística del Recolector de Basura. Es tan desaconsejable que hasta hay una opción en la máquina virtual para desactivar estas llamadas: -XX:+DisableExplicitGC
En consecuencia, el único inconveniente a este mecanismo es determinar cuándo se tiene que ejecutar el recolector de basura. Existen varios algoritmos para hacerlo, pero el más eficiente es el primero de ellos:

1. Esperar a que no quede memoria libre, y entonces, ejecutar el recolector de basura.
2. Fijar un umbral de ocupación de la memoria libre y ejecutar el recolector de basura cuando se supere dicho umbral.
3. Ejecutar el recolector de basura a intervalos regulares (no siempre es posible).
4. Ejecutar el recolector de basura justo antes de cada reserva de memoria.
5. Permitir al programador que invoque explícitamente al recolector de basura cuando quiera.

Hay que recordar que el GC reclama las reservas de memoria que se producen en el programa no importando el porcentaje que ocupe este en el CPU por ende si tuviéramos el parámetro ParallelGCThreads configurado con 2 ó con 4 procesadores estos van a cumplir con la tarea de reclamar las reservas de memoria el punto clave es de que están realizando un esfuerzo mucho mayor (50% para ser exactos) comparado con el trabajo

que realizarían los 8 procesadores correctamente configurado del parámetro ParallelGCThreads.

Cabe resaltar que leí y analice la documentación correspondiente a la configuración de CMM y no encontré restricción alguna acerca de la configuración de los procesadores del GC de la aplicación. Una vez que la JVM esté debidamente configurada se seguirá monitoreando la aplicación para evitar demasiada concurrencia en los métodos y clases. Además se dará una mejor alternativa para el mejoramiento de las buenas prácticas de desarrollo y así evitar código repetido, no liberación de memoria de los objetos y no cierre de conexiones a bases de datos.

ANEXOS

Diseño e implementación de los scripts FTP_External_TS.sh y FTP_Internal_TS.sh

Se requiere hacer el diseño de dos scripts llamados FTP_External_TS.sh y FTP_Internal_TS.sh para realizar pagos externos e internos.

Proceso para su solución:

1. Hacer un análisis técnico del flujo de información y generar el documento con dicho análisis.
2. Hacer los scripts.
3. Implementación de scripts
4. Generar el ITG para correr pruebas en el ambiente de desarrollo.
5. Si las pruebas son exitosas en el ambiente de desarrollo, hacer la implementación en Calidad y correr pruebas.
6. Si las pruebas son exitosas en el ambiente de Calidad, hacer la implementación en Producción y correr pruebas.

Análisis Técnico

WSS ICL 7.1 CMM
Este es el flujo del proceso a ejecutar cuando se requiera hacer un pago en WSS ICL

WSS ICL:

1. En WSS ICL CMM, el usuario accesa a

Payment Factory/Transaction Workflow/Release Payments

Y selecciona los pagos a enviar

2. Internamente el proceso que se ejecuta es:

a) El Export controller es invocado
b) De acuerdo al pago seleccionado el mecanismo de CMM Customized Transport se ejecutara.

FTP_External_TS.sh
FTP_Internal_TS.sh
FTP_External.sh
FTP_Internal.sh

Luego se ejecuta lo siguiente:

a) El pago generado de CMM es renombra con extensión txt.xml.
b) Los archivos .xml son encriptados usando PGP y el archive generado es txt.xml.pgp.
c) Una vez encriptado el archivo este es enviado a pgp farm
d) Los archivos generados son removidos al folder de ARCHIVE.

PGP farm:

Del lado de PGP hay una tarea que se ejecuta para encriptar los pagos una vez que estos han sido enviados.

a) La tarea se llama pgp-trema & pgp-tremaext
b) Cuando el archivo es encontrado este es decriptado.
c) El archivo decriptado es enviado a webcash.
d) Un script es ejecutado para cargar los archivos a Webcash
c) Se envia una notificación via correo electrónico

Webcash:

Una vez que el archivo ha sido decriptado

a) El archivo es exportado a webcash.

El diagrama de la fig. 6.1 explica mejor el proceso.

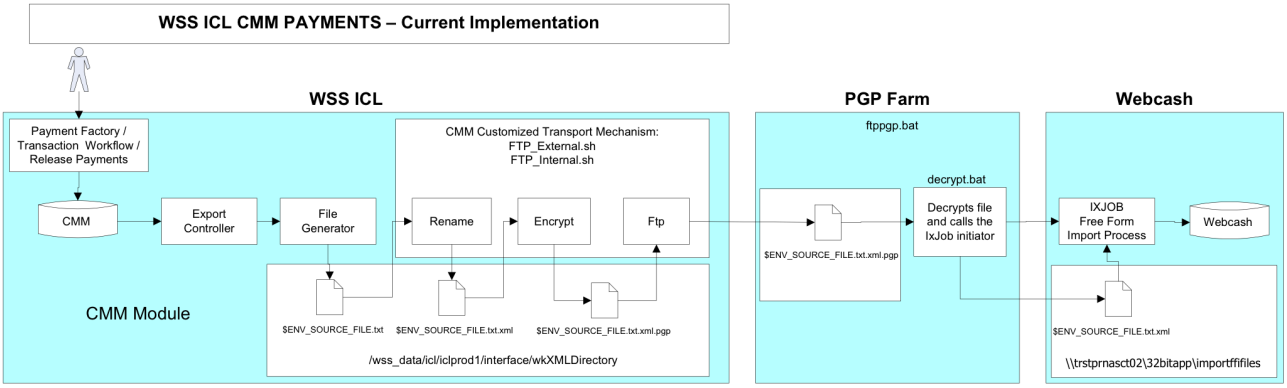


fig. 6.1 WSS ICL Payments Forecasting

En la fig. 6.2 la actividad TRM ha sido implementada con la finalidad de generar pagos

a) El usuario accesa a TRM Manager Board y programa la actividad TRM-PAYMENT-FORECAST.
b) La actividad es ejecutada y se genera el pago en samba

c) El usuario obtiene el pago de samba y lo mueve manualmente al servidor de webcash.

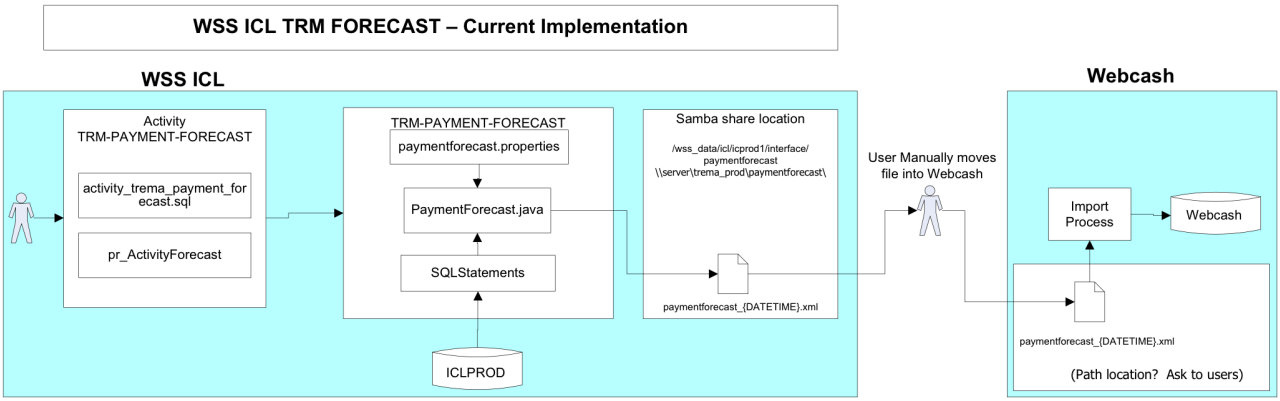


Fig 6.2 WSS ICL 7.1 Payments Forecasting – Implementación propuesta

Por ende el flujo será;

a) El usuario accesa a TRM Manager Board y programa la actividad TRM-PAYMENT-FORECAST.
b) La actividad es ejecutada y se genera el pago en samba
c) Un nuevo script es ejecutado para encriptar y enviar el pago a webcash.
d) PGP necesita modificarse para decriptar el pago y moverlo al servidor de webcash.

Diagrama 6.3:

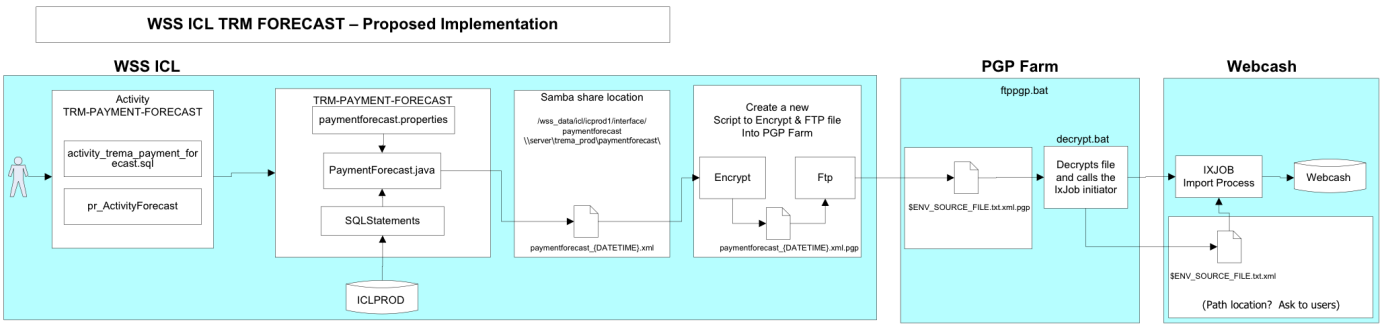


Fig. 6.3 Implementación propuesta de TRM Forecast

Diseño de Scripts.

Se procede a hacer el diseño de los scripts

FTP_External_TS.sh
FTP_Internal_TS.sh

Impact Analysis Document

General Information	
Kintana Request ID:	
Application: WSS-ICF	
Change Name: Front end side On the view side we have to add Ftp_External_Trust.sh and Ftp _Internal_Trust.sh Back end side We have to create two shell files they will be named Ftp _External_Trust.sh and Ftp _Internal_Trust.sh which are going to send the file encrypted to PGP farm.	
Team	
Role	Name
Technical Analyst	César Saucedo / Armando Rodriguez
Project Manager	
Application Support	César Saucedo / Armando Rodriguez
Problem / Current Situation	

Front end side

1. First of all the user selects the payments to be sent to PGP farm

Currently the there are a list of the scripts

- FtpExportExternal
- FtpExportInternal

- xmlProcess
- None Selected

Back end Side

2. After the payment to send is selected by the user, following processes are executed:

a) Export controller is invoked

b) According selected payment to be released a file is generated in location

/…/wkXMLDirectory as $ENV_SOURCE_FILE.txt

depending the payment selected, following customized shell script is executed:

FTP_External.sh
FTP_Internal.sh

And following steps are executed:

a) Generated CMM export .txt file is renamed as a .xml file, e.g.,
/…/BANK080916172011001.txt
is renamed as
/…/BANK080916172011001.txt.xml

b) .xml files are encrypted using pgp tool: encrypted .xml.pgp file is generated, e.g.,
/…/BANK080916172011001.txt.xml.pgp

c) encrypted .xml.pgp file is ftp into pgp farm

d) generated files are moved into ARCHIVE directory, e.g.,
/…/BANK080916172011001.txt

Summary of Proposed Solution / Change

Front end side

1. First of all the user selects the payments to be sent to PGP farm

Add to the list Ftp_External_Trust.sh and Ftp_Internal_Trust.sh.

- FtpExportExternal
- FtpExportInternal
- Ftp_External_Trust.sh
- Ftp_Internal_Trust.sh
- xmlProcess
- None Selected

Back end Side

2. After the payment to send is selected by the user, following processes are executed:

a) Export controller is invoked

b) According selected payment to be released a file is generated in location

depending the payment selected, following customized shell script is executed:

Add the scripts Ftp_External_Trust.sh and Ftp_Internal_Trust.sh.

Ftp_External.sh
Ftp_Internal.sh
Ftp_External_Trust.sh
Ftp_Internal_Trust.sh

And following steps are executed:

a) Generated CMM export .txt file is renamed as a .xml file

b) .xml files are encrypted using pgp tool: encrypted .xml.pgp file is generated

c) encrypted .xml.pgp file is ftp into pgp farm

d) generated files are moved into ARCHIVE directory

Release Document

System Release Document
ITG 79492 - Business Requirements – Implementation of FTP_External_TS script and FTP_Internal_TS script

Document Purpose
This document lists all the steps that need to be taken to release the system in to QA and production locations. The release team, tech lead and the BA review this document during the release meeting and make sure that everyone understands the release requirements and that everyone is in agreement. The end goal of this document is to provide a step-by-step guide for the release team so that the release to QA and production are performed the same way and that no steps are missed.

Release Owner		Swati Mehta			
Release Resources:					
DBA:	NA	WebMaster:	NA	Sys	NA

				Admin:	
Crystal:	NA	Mainframe:	N/A		
Other [DWH]:	NA				
Release Support Resources:					
Tech Lead:	Cesar Saucedo/ Armando Rodriguez	PM/BA:	Swati Mehta		
User Acceptance Tester:		NA			

Production Release Constraints:

Release Description:

These steps are to create two scripts files they will name Ftp _External_TS.sh and Ftp _Internal_TS.sh which will send the payment external or internal encrypted to PGP farm.

Release notes

New features
Add two scripts

Defects or bugs fixed
N/A
General Notes
N/A

Impact on monitoring tools
N/A

Impact on Data Warehouse
NA

Impact on Disaster Recovery
N/A

Impact on Off site systems
N/A

Test results

Reference Documents:
[INSTRUCTIONS: List or attached relevant reference documents as they support or pertain to this release.]

Proposed Change:

Revision: 1
Function: Build
Product Owner:
Developers : Cesar Saucedo / Armando Rodriguez
Environment : PROD
Date :

Installation process steps

Unix Admin tier

#	During release to QA TRSTSQAWSSAPP01	During Production release trstsprwssapv01	During DR release TRALSDRWSSAPV01
1	Copy the Scripts (log in as fkbin user) Copy the files below: FTP_External_TS.sh FTP_Internal_TS.sh XMLPaymentEXTTS.txt XMLPaymentINTTS.txt from DEV2 environment its hostname is TRSTSQATREMDB01	Copy the Scripts (log in as fkbin user) Copy the files below: FTP_External_TS.sh FTP_Internal_TS.sh These resources will be sent by email, the folder will be named ProductionResources and it will has three folders respectively: ProductionScripts ProductionKeys ProductionPassphrase ProductionScripts.- This folder will contain the FTP_External_TS.sh and the FTP_Internal_TS.sh scripts respectively. ProductionKeys.- This folder will contain the WSSFFTREMAEXTTS.asc, WSSFFTREMAEXTTS_PU.asc WSSFFTREMATS.asc WSSFFTREMATS_PU .asc keys respectively.	Copy the Scripts (log in as fkbin user) Copy the files below: FTP_External_TS.sh FTP_Internal_TS.sh These resources will be sent by email, the folder will be named ProductionResources and it will has three folders respectively: ProductionScripts ProductionKeys ProductionPassphrase ProductionScripts.- This folder will contain the FTP_External_TS.sh and the FTP_Internal_TS.sh scripts

		ProductionPassphrase.- This folder will contain the XMLPaymentEXTTS.txt XMLPaymentINTTS.txt respectively.	respectively. ProductionKeys.- This folder will contain the WSSFFTREMAEXTTS.asc, WSSFFTREMAEXTTS_PU.asc WSSFFTREMATS.asc WSSFFTREMATS_PU .asc keys respectively. ProductionPassphrase.- This folder will contain the XMLPaymentEXTTS.txt XMLPaymentINTTS.txt respectively.
2	Paste the Scripts (log in as fkbin user) Paste the files above mentioned to QA2 …/CMM	Paste the Scripts (log in as fkbin user) Paste the files FTP_External_TS.sh FTP_Internal_TS.sh below path …/CMM	Paste the Scripts (log in as fkbin user) Paste the files FTP_External_TS.sh FTP_Internal_TS.sh below path …/CMM
3	Grant permission (log in as fkbin user) Grant permissions to the scripts as below chmod 777 FTP_External_TS.sh chmod 777 FTP_Internal_TS.sh	Copy the Phase phrase (log in as fkbin user) Copy the files below from the ProductionResources folder : XMLPaymentEXTTS.txt XMLPaymentINTTS.txt	Copy the Phase phrase (log in as fkbin user) Copy the files below from the ProductionResources folder : XMLPaymentEXTTS.txt XMLPaymentINTT

			S.txt
4	Copy the Security Keys (log in as fkbin user) Copy the files below: WSS FFTREMAEXTTS_Pri.asc WSS FFTREMAEXTTS_Public.asc WSS FFTREMATS_Pri.asc WSS FFTREMATS_Public.asc from DEV2 environment its hostname is TRSTSQATREMDB01 The path where the files are allocated is: …/key	Paste the Pass phrase (log in as fkbin user) Paste the files XMLPaymentEXTTS.txt XMLPaymentINTTS.txt below path …/CMM	Paste the Pass phrase (log in as fkbin user) Paste the files XMLPaymentEXTTS.txt XMLPaymentINTTS.txt below path …/CMM
5	Paste the Security Keys (log in as fkbin user) Create a new folder named key and paste the files above mentioned to QA2 inside the key folder, the path should be …/key	Grant permission (log in as fkbin user) Grant permissions to the scripts as below chmod 777 FTP_External_TS.sh chmod 777 FTP_Internal_TS.sh	Grant permission (log in as fkbin user) Grant permissions to the scripts as below chmod 777 FTP_External_TS.sh chmod 777 FTP_Internal_TS.sh
6	Import the Security Keys (log in as fkbin user) To import the public and private security keys you have to execute the next four commands. bash-3.00# …/gpg --import 'WSS FFTREMAEXTTS_Pri.asc'	Copy the Security Keys (log in as fkbin user) Copy the files below from the ProductionResources folder : WSSFFTREMAEXTTS.asc WSSFFTREMAEXTTS_PU.asc WSSFFTREMATS.asc WSSFFTREMATS_PU .asc	Copy the Security Keys (log in as fkbin user) Copy the files below from the ProductionResources folder : WSSFFTREMAEXTTS.asc

	Then the System is going to ask you next. Really update the preferences? (y/N) The answer must be: Y Finally the system is going to ask you next: Enter passphrase: You must type the passphrase given by the security department. bash-3.00# …/gpg --import ' WSS FFTREMAEXTTS_Public.asc ' Then the System is going to ask you next. Really update the preferences? (y/N) The answer must be: Y *For public key passphrase is not necessary. bash-3.00# …/gpg --import ' WSS FFTREMATS_Pri.asc ' Then the System is going to ask you next. Really update the preferences? (y/N) The answer must be: Y Finally the system is going to ask you next: Enter passphrase: You must type the passphrase given by the security department.		WSSFFTREMAEXTTS_PU.asc WSSFFTREMATS.asc WSSFFTREMATS_PU .asc

	bash-3.00# …/gpg --import ' WSS FFTREMATS_Public.asc ' Then the System is going to ask you next. Really update the preferences? (y/N) The answer must be: Y *For public key passphrase is not necessary.		
		Paste the Security Keys Create a new folder named key and paste the files above mentioned to Production inside the key folder, the path should be …/key	Paste the Security Keys Create a new folder named key and paste the files above mentioned to DR inside the key folder, the path should be …/key
		Import the Security Keys (log in as fkbin user) To import the public and private security keys you have to execute the next four commands. bash-3.00# …/gpg --import ' WSSFFTREMAEXTTS.asc ' Then the System is going to ask you next. Really update the preferences? (y/N) The answer must be: Y Finally the system is going to	Import the Security Keys (log in as fkbin user) To import the public and private security keys you have to execute the next four commands. bash-3.00# …/gpg --import ' WSSFFTREMAEXTTS.asc ' Then the System is going to ask you next.

		ask you next: Enter passphrase: You must type the passphrase given by the security department. bash-3.00# …/gpg --import ' WSSFFTREMAEXTTS_PU.asc ' Then the System is going to ask you next. Really update the preferences? (y/N) The answer must be: Y *For public key passphrase is not necessary. bash-3.00# …/gpg --import ' WSSFFTREMATS.asc ' Then the System is going to ask you next. Really update the preferences? (y/N) The answer must be: Y Finally the system is going to ask you next: Enter passphrase: You must type the passphrase given by the security department. bash-3.00# …/gpg --import ' WSSFFTREMATS_PU .asc ' Then the System is going to ask you next. Really update the	Really update the preferences? (y/N) The answer must be: Y Finally the system is going to ask you next: Enter passphrase: You must type the passphrase given by the security department. bash-3.00# …/gpg --import ' WSSFFTREMAEXTTS_PU.asc ' Then the System is going to ask you next. Really update the preferences? (y/N) The answer must be: Y *For public key passphrase is not necessary. bash-3.00# …/gpg --import ' WSSFFTREMATS.asc ' Then the System is going to ask you next. Really update the preferences? (y/N) The answer must be: Y

		preferences? (y/N) The answer must be: Y *For public key passphrase is not necessary.	Finally the system is going to ask you next: Enter passphrase: You must type the passphrase given by the security department. bash-3.00# …/gpg --import ' WSSFFTREMATS _PU .asc ' Then the System is going to ask you next. Really update the preferences? (y/N) The answer must be: Y *For public key passphrase is not necessary.

Application Support tier

#	During release to QA TRSTSQAWSSAPP01	During Production release trstsprwssapv01	During DR release TRALSDRWSSAPV01
1	Wait the Unix Admin team finish their part above mentioned.	Wait the Unix Admin team finish their part above mentioned.	Wait the Unix Admin team finish their part above mentioned.
2	Go to the CMM QA2 website and log in.	Go to the CMM Production website and log in.	Go to the CMM DR website and log in.
3	Inside the web application above mentioned In the main menu go to: …/ Communication Protocol	Inside the web application above mentioned In the main menu go to: …/ Communication Protocol	Inside the web application above mentioned In the main menu go to:

	Parameters	Parameters	…/ Communication Protocol Parameters
4	Choose Generic Command Line Communication option and click edit parameter button.	Choose Generic Command Line Communication option and click edit parameter button.	Choose Generic Command Line Communication option and click edit parameter button.
5	Click in the new entry button.	Click in the new entry button.	Click in the new entry button.
6	FtpExportExternalTS To add the FtpExportExternalTS follow type next in the fields: Field: Parameter Set Name Value: FtpExportExternalTS Field: Command Line Value: …/CMM/FTP_External_TS.sh Field: Local Path Value: …/wkXMLDirectory Field: Remote Path Value: / Field: File Filters Value: N/A Field: Enabled Value: Yes Field: Time Out Value (milliseconds) Value: 500,000 Field: Action on Local Files Value: No Action	FtpExportExternalTS To add the FtpExportExternalTS follow type next in the fields: Field: Parameter Set Name Value: FtpExportExternalTS Field: Command Line Value: …/CMM/FTP_External_TS.sh Field: Local Path Value: …/wkXMLDirectory Field: Remote Path Value: / Field: File Filters Value: N/A Field: Enabled Value: Yes Field: Time Out Value (milliseconds) Value: 500,000 Field: Action on Local Files Value: No Action	FtpExportExternalTS To add the FtpExportExternalTS follow type next in the fields: Field: Parameter Set Name Value: FtpExportExternalTS Field: Command Line Value: …/CMM/FTP_External_TS.sh Field: Local Path Value: …/wkXMLDirectory Field: Remote Path Value: / Field: File Filters Value: N/A Field: Enabled Value: Yes Field: Time Out Value (milliseconds) Value: 500,000 Field: Action on Local Files

			Value: No Action
7	FtpExportInternalTS To add the FtpExportExternalTS follow type next in the fields: Field: Parameter Set Name Value: FtpExportInternalTS Field: Command Line Value: …/CMM/FTP_External_TS.sh Field: Local Path Value: …/wkXMLDirectory Field: Remote Path Value: / Field: File Filters Value: N/A Field: Enabled Value: Yes Field: Time Out Value (milliseconds) Value: 500,000 Field: Action on Local Files Value: No Action	FtpExportInternalTS To add the FtpExportExternalTS follow type next in the fields: Field: Parameter Set Name Value: FtpExportInternalTS Field: Command Line Value: …/CMM/FTP_External_TS.sh Field: Local Path Value: …/wkXMLDirectory Field: Remote Path Value: / Field: File Filters Value: N/A Field: Enabled Value: Yes Field: Time Out Value (milliseconds) Value: 500,000 Field: Action on Local Files Value: No Action	FtpExportInternalTS To add the FtpExportExternalTS follow type next in the fields: Field: Parameter Set Name Value: FtpExportInternalTS Field: Command Line Value: …/CMM/FTP_External_TS.sh Field: Local Path Value: …/wkXMLDirectory Field: Remote Path Value: / Field: File Filters Value: N/A Field: Enabled Value: Yes Field: Time Out Value (milliseconds) Value: 500,000 Field: Action on Local Files Value: No Action

Special notes
- N/A

Rollback procedures

Unix Admin tier rollback steps

#	During release to QA1	During Production release	During DR release
1	Delete the Scripts (log in as fkbin user) Delete the scripts FTP_External_TS.sh FTP_Internal_TS.sh XMLPaymentEXTTS.txt XMLPaymentINTTS.txt from the path below …/CMM	Delete the Scripts and the phase phrase (log in as fkbin user) Delete the scripts and the phase phrases FTP_External_TS.sh FTP_Internal_TS.sh XMLPaymentEXTTS.txt XMLPaymentINTTS.txt from the path below …/CMM	Delete the Scripts and the phase phrase (log in as fkbin user) Delete the scripts and the phase phrases FTP_External_TS.sh FTP_Internal_TS.sh XMLPaymentEXTTS.txt XMLPaymentINTTS.txt from the path below …/CMM
2	Delete the Security Keys from the QA2 box (log in as fkbin user) To delete the public and private security keys you have to execute the next command bash-3.00# …/gpg --delete-secret-keys 'WSSFFTREMAEXTTS' bash-3.00# …/gpg --delete-secret-keys 'WSSFFTREMATS' Make sure there is no keys executing command below. -bash-3.00$ …/gpg --list-keys	Delete the Security Keys from the production box (log in as fkbin user) To delete the public and private security keys you have to execute the next command bash-3.00…/gpg --delete-secret-keys 'WSSFFTREMAEXTTS' bash-3.00# …/gpg --delete-secret-keys 'WSSFFTREMATS' Make sure there is no keys executing command below. -bash-3.00…/gpg --list-keys	Delete the Security Keys from the production box (log in as fkbin user) To delete the public and private security keys you have to execute the next command bash-3.00# …/gpg --delete-secret-keys 'WSSFFTREMAEXTTS' bash-3.00# …/gpg --delete-secret-keys 'WSSFFTREMATS' Make sure there is no keys executing command below. -bash-3.00$ …/gpg --list-keys

3	Delete the key folder (log in as fkbin user) Delete the key folder which is in …/key and has the Security Keys WSS FFTREMAEXTTS_Pri.asc WSS FFTREMAEXTTS_Public.asc WSS FFTREMATS_Pri.asc WSS FFTREMATS_Public.asc	Delete the key folder (log in as fkbin user) Delete the key folder which is in …/key and has the Security Keys WSSFFTREMAEXTTS.asc WSSFFTREMAEXTTS_PU.asc WSSFFTREMATS.asc WSSFFTREMATS_PU .asc	Delete the key folder (log in as fkbin user) Delete the key folder which is in …/key and has the Security Keys WSSFFTREMAEXTTS.asc WSSFFTREMAEXTTS_PU.asc WSSFFTREMATS.asc WSSFFTREMATS_PU .asc

Application Support tier rollback steps

#	During release to QA	During Production release	During DR release
1	Wait the Unix Admin team finish their part above mentioned.	Wait the Unix Admin team finish their part above mentioned.	Wait the Unix Admin team finish their part above mentioned.
2	Go to the CMM QA2 website and log in.	Go to the CMM Production website and log in.	Go to the CMM DR website and log in.
3	Inside the web application above mentioned In the main menu go to: … / Communication Protocol Parameters	Inside the web application above mentioned In the main menu go to: … / Communication Protocol Parameters	Inside the web application above mentioned In the main menu go to: …/ Communication Protocol Parameters
4	Choose Generic Command Line Communication option and click edit parameter button.	Choose Generic Command Line Communication option and click edit parameter button.	Choose Generic Command Line Communication option and click edit parameter button.
5	Choose FtpExportExternalTS and click on the Delete button.	Choose FtpExportExternalTS and click on the Delete button.	Choose FtpExportExternalTS and click on the

			Delete button.
6	Choose FtpExportInternalTS and click on the Delete button.	Choose FtpExportInternalTS and click on the Delete button.	Choose FtpExportInternalTS and click on the Delete button.

Additional information required by object type

New Package

New Table

New Database

New Scheduled Task

5.0 Master List of Objects to be installed

Estimated Install time: 45 mins.

Server: ICLPROD

Database: ICL

Other:

Resultados

Se corren pruebas en Desarrollo con éxito
Se corren pruebas en Calidad con éxito
Se corren pruebas en Producción con éxito.

Actualización de código para concatenar el owner ID

Problema:

Se requiere actualizar código para la concatenación de un owner ID

Proceso para su solución:

1. Se requiere hacer un análisis técnico del flujo de información.
2. Se genera la documentación para codificar e implementar el código posterior a esto se corren pruebas en el ambiente de desarrollo.
3. Si las pruebas son exitosas en el ambiente de desarrollo, hacer la implementación en Calidad y correr pruebas.
4. Si las pruebas son exitosas en el ambiente de Calidad, hacer la implementación en Producción y correr pruebas.

Análisis

Se requiere concatenar el ID owner a cada entry ID de la actividad MARS y enviar la notificación por correo.

Activiad TRM-toMARS

Activity Name	Output	Location of Output
TRM-TO-MARS	MARSGL.TREMA.VALID.csv	/…/archive

SAMPLE of “TO BE” mail generated with updates
Ejemplo:
Hi, in regards the analysis on activity TRM-TO-MARS for the warning message “0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (ENTRY ID'S) 2940525(OWNER ID), 2958055 (OWNER ID), 3567419(OWNER ID), 3644554(OWNER ID), 3987482(OWNER ID), 3987484 (OWNER ID), 4143639(OWNER ID), 4143641(OWNER ID), 4570578(OWNER ID), 4570580(OWNER ID)) doesnt have all information required and MARS validation probably will fail”

QUERY

```
select id, account, bsla, le, intercompany_le, ctpy, ctpy_bsla, Type, currency_id from  TMPMARSEN
where account is null or bsla is null or le is null or
intercompany_le is null or ctpy is null or ctpy_bsla is null or Type is null
or currency_id  is null
order by ID
```

Generación de documentación.

Release Document

ITG 77423- Business Requirements – Update of Java classes to update the email generated by TRM-TO-MARS activity.

Document Purpose
This document lists all the steps that need to be taken to release the system in to QA and production locations. The release team, tech lead and the BA review this document during the release meeting and make sure that everyone understands the release requirements and that everyone is in agreement. The end goal of this document is to provide a step-by-step guide for the release team so that the release to QA and production are performed the same way and that no steps are missed.

Release Owner					
Release Resources:					
DBA:	NA	WebMaster:	NA	Sys Admin:	NA
Crystal:	NA	Mainframe:	N/A		
Other [DWH]:	NA				
Release Support Resources:					
Tech Lead:		PM/BA:			
User Acceptance Tester:		NA			

Production Release Constraints:

Release Description:

These steps are to add owner ID next to each entry ID found after execution of MARS Extract activity within WSS-ICL for the system generated e-mail generated after successful completion of activity.

1. Release notes

    1.1. New features
        - Add of the owner_id field
        - Add a string and the current date and time.
    1.2. Defects or bugs fixed

N/A

    1.3. General Notes

N/A

    1.4. Impact on monitoring tools

N/A

1.5. Impact on Data Warehouse

NA

1.6. Impact on Disaster Recovery

N/A

1.7. Impact on Off site systems

N/A

1.8. Test results

Reference Documents:
[INSTRUCTIONS: List or attached relevant reference documents as they support or pertain to this release.]

Proposed Change:
Revision: 1

Function: Build
Product Owner:
Developers : WSS Consultant
Environment : PROD
Date :

2. Installation process steps

2.1. Unix Admin tier

#	During release to QA	During Production release	During DR release
1	Get the MARSBalance.java, MARSFile.java, MARSFile.properties and SQLStatements.java classes and make a backup of them	Get the MARSBalance.java, MARSFile.java, MARSFile.properties and SQLStatements.java classes and make a backup of them	Get the MARSBalance.java, MARSFile.java, MARSFile.properties and SQLStatements.java classes and make a backup of them
2	Copy the MARSBalance.java from Dev2: …/mars	Copy the MARSBalance.java from Dev2: …/mars	Copy the MARSBalance.java from Dev2: …/mars
3	QA	Production	Production

	Paste the MARSBalance.java gotten in the next path: …/mars	Paste the MARSBalance.java gotten in the next path: …/mars	Paste the MARSBalance.java gotten in the next path: …/mars
4	Copy the SQLStatements.java from Dev2: …/mars	Copy the SQLStatements.java from Dev2: …/mars	Copy the SQLStatements.java from Dev2: …/mars
5	QA Paste the SQLStatements.java gotten in the next path: …/mars	Production Paste the SQLStatements.java gotten in the next path: …/mars	Production Paste the SQLStatements.java gotten in the next path: …/mars
6	Copy the MARSFile.java from Dev2: …/mars	Copy the MARSFile.java from Dev2: …/mars	Copy the MARSFile.java from Dev2: …/mars
7	QA Paste the MARSFile.java gotten in the next path: …/mars	Production Paste the MARSFile.java gotten in the next path: …/mars	Production Paste the MARSFile.java gotten in the next path: …/mars
8	Open MARSFile.properties which is in …/mars Update next line info.fileCreated=\n* rows reported. \nFile created correctly. By info.fileCreated=\n* . \nFile created correctly. Save the file.	Open MARSFile.properties which is in …/mars Update next line info.fileCreated=\n* rows reported. \nFile created correctly. By info.fileCreated=\n* . \nFile created correctly. Save the file.	Open MARSFile.properties which is in …/mars Update next line info.fileCreated=\n* rows reported. \nFile created correctly. By info.fileCreated=\n* . \nFile created correctly. Save the file.

1.1. Special notes

- N/A

2. Rollback procedures

2.1. Unix Admin tier rollback steps

#	During release to QA1 TRSTWPRCTXAPP03	During Production release trstsprwssapv01	During DR release TRALSDRWSSAPV01
1	Get the MARSBalance.java, MARSFile.java, MARSFile.properties and SQLStatements.java backup.	Get the MARSBalance.java and SQLStatements.java classes backup.	Get the MARSBalance.java and SQLStatements.java classes backup.
2	Copy the MARSBalance.java backup	Copy the MARSBalance.java backup	Copy the MARSBalance.java backup
3	Paste the MARSBalance.java gotten in the next path: …/mars	Paste the MARSBalance.java gotten in the next path: …/mars	Paste the MARSBalance.java gotten in the next path: …/mars
4	Copy the SQLStatements.java:	Copy the SQLStatements.java:	Copy the SQLStatements.java:
5	Paste the SQLStatements.java gotten in the next path: …/mars	Paste the SQLStatements.java gotten in the next path: …/mars	Paste the SQLStatements.java gotten in the next path: …/mars
6	Copy the MARSFile.java	Copy the MARSFile.java	Copy the MARSFile.java
7	Paste the SQLStatements.java gotten in the next path: …/ge/	Paste the SQLStatements.java gotten in the next path: …/ge/	Paste the SQLStatements.java gotten in the next path: …/ge/
8	Copy the MARSFile.properties	Copy the MARSFile.properties	Copy the MARSFile.properties
9	Paste the SQLStatements.java gotten in the next path: …/ge/	Paste the MARSFile.properties gotten in the next path: …/ge/	Paste the MARSFile.properties gotten in the next path: …/ge/

4.Additional information required by object type

4-1 New Package

4.2 New Table
4.3 New Database
4.4 New Scheduled Task

5.0 Master List of Objects to be installed

Estimated Install time: 45 mins.

Server: ICLPROD

Database: ICL

Other:

Impact Analysis Document

General Information	
Kintana Request ID: 77423	
Application: WSS-ICF	
Change Name: 1. To add owner ID next to each entry ID found after execution of MARS Extract activity within WSS-ICL for the system generated e-mail generated after successful completion of activity. 2. At the end of the email there is a string which has to be modified.	
Team	
Role	Name
Technical Analyst	Armando Rodriguez
Project Manager	
Application Support	Armando Rodriguez
Problem / Current Situation	
• When MARS Extract finishes its activity successfully an email is generated, it is formed by Exceptions and the Entry ID, so it needs the Owner ID concatenated. • The string at the end of the email generated “44784 rows reported” was updated by “44784 Transactions exported on 09/23/2010 12:08:44”	
Summary of Proposed Solution / Change	

- Class MARSBalance.java

Current Java Code (line 103)

```
while (rs.next()) {
      java.math.BigDecimal vEntryID = rs.getBigDecimal("id");
```

```
      strExceptions=strExceptions+" "+vEntryID.toString();
     }
```

Update Java Code (line 103) like this.

```
   while (rs.next()) {
java.math.BigDecimal vEntryID = rs.getBigDecimal("id");
          String ownerID = rs.getString("owner");
           if(ownerID != null) {
                strExceptions=strExceptions+" "+vEntryID.toString() + " " + "("+ownerID+")";

           }

      }
```

-Class MARSFile.java

Current Java Code (line 97)

```
if (writer.writeFile(resBundle, writerModel, new String[] { eDate }) == 1) {
         rowsSelected = ((ArrayList) writerModel.get("data")).size();
        appLog.info(resBundle.getString("info.fileCreated").replaceAll("\\*",
String.valueOf(rowsSelected));
     }
```

Update Java Code (line 97) like this.

```
if (writer.writeFile(resBundle, writerModel, new String[] { eDate }) == 1) {

         rowsSelected = ((ArrayList) writerModel.get("data")).size();
         SimpleDateFormat dateTimeFormat = new SimpleDateFormat("MM/dd/yyyy
HH:mm:ss");
         Calendar calendar = Calendar.getInstance();

appLog.info(resBundle.getString("info.fileCreated").replaceAll("\\*",
String.valueOf(rowsSelected).concat(" Transactions exported on

").concat((dateTimeFormat.format(calendar.getTime()))));

     }
```

- Class SQLStatements.java

Current Java Code (line 217)

```
select id, account, bsla, le, intercompany_le, ctpy, ctpy_bsla, Type, currency_id
from  TMPMARSENTRYSELECTION
where
account is null
or bsla is null
or le is null
or intercompany_le is null
or ctpy is null
```

or ctpy_bsla is null
or Type is null
or currency_id is null
order by ID ;

Update Java Code (line 217)
select id, account,
bsla, le, intercompany_le,

ctpy, ctpy_bsla, Type,
currency_id, owner

Change (for each change, specify detected systems impact below)	Details	
CHANGE	DESCRIPTION	
Update the java code and compile the MARSBalance.java class	The java code must be updated in order to add the owner id like it is required.	
Update the java code and compile the MARSFile.java class	The java code must be updated in order to add a string and the current date & time like it is required.	
Update the java code and compile the SQLStatements.java class	The query must be updated in order to add owner id column like it is required.	
Detected Systems Impact (to other objects, applications, database structure, DW, APC, DR Plans, Security, etc.)		
IMPACT	ACTIONS REQUIRED	RESPONSIBLE
Change java code	Run the Activity TRM-TO-MARS check the mail and the owner id concatenated	Armando Rodriguez
Benefit Impact	The application will perform properly.	
Benefit Assumptions:		
Approval Status:		
Approval Status:	☐ Approved ☐ Denied ☐ Deferred	
Approved By:		

Robo Help Document

ITG change #: 77423
Issue/Problem:

The e-mail generated by TRM-TO-MARS activity requires the OwnerID concatenated next to entryID.

Application Functional Area: WSS-ICL

Key Words: entryID, ICL, OwnerID.

Symptoms/ Manifestations

These changes are required to get more information about the exceptions.

Resolution Steps:

7.1.5.5a Hotfix for TRM-TO-MARS

Application Support Tier

QA1 Environment

1. Load environment variables.

2. Back-up next files.

MARSBalance.class
File: /mars/MARSBalance.class
From: /mars/MARSBalance.class
To: /mars/MARSBalance.(YYYYMMDD).class

SQLStatements.class
File: /mars/SQLStatements.class
From: /mars/SQLStatements.class
To: /mars/SQLStatements.(YYYYMMDD).class

MARSFile.class
File: /mars/MARSFile.class
From: /ic/mars/MARSFile.class
To: /mars/MARSFile.(YYYYMMDD).class

MARSFile.properties
File: /ic/mars/MARSFile.properties
From: /ic/mars/MARSFile.properties
To: /ic/mars/MARSFile(YYYYMMDD).properties

3. Update next files.

The MARSBalance.class file must be updated from DEV2 environment to QA1 environment
that is because it is already updated and compiled.

MARSBalance.class
File: /mars/MARSBalance.class
From /ic/mars/
To: /mars/

The SQLStatements.class file must be updated from DEV2 environment to QA1 environment
that is because it is already updated and compiled.

SQLStatements.class
File: /mars/SQLStatements.class
From: /ic/mars/
To: /mars/

The MARSFile.class file must be updated from DEV2 environment to QA1 environment
that is because it is already updated and compiled.

MARSFile.class
File: /mars/MARSFile.class
From: /ic/mars/
To: /mars/

The MARSFile.properties file must be updated from DEV2 environment to QA1 environment
that is because it is already updated.

MARSFile.class
File: /mars/MARSFile.properties
From: /ic/mars/
To: /mars/

4. Run the TRM-TO-MARS Activity from de Activity Manager.

Workaround:
DB Admin, Application Support team and Middleware Team for accuracy.

Escalation:
César Saucedo (Softtek), Armando Rodríguez (Softtek

Release Document

System Release Document
ITG 77423- Business Requirements – Update of Java classes to update the email generated by TRM-TO-MARS activity.

Document Purpose
This document lists all the steps that need to be taken to release the system in to QA and production locations. The release team, tech lead and the BA review this document during the release meeting and make sure that everyone understands the release requirements and that everyone is in agreement. The end goal of this document is to provide a step-by-step guide for the release team so that the release to QA and production are performed the same way and that no steps are missed.

Release Owner					
Release Resources:					
DBA:	NA	WebMaster:	NA	Sys Admin:	NA
Crystal:	NA	Mainframe:	N/A		
Other [DWH]:	NA				
Release Support Resources:					
Tech Lead:		PM/BA:			
User Acceptance Tester:		NA			

Production Release Constraints:

Release Description:

These steps are to add owner ID next to each entry ID found after execution of MARS Extract activity within WSS-ICL for the system generated e-mail generated after successful completion of activity.

3. Release notes

3.1. New features
- Add of the owner_id field
- Add a string and the current date and time.

3.2. Defects or bugs fixed

N/A

3.3. General Notes

N/A

3.4. Impact on monitoring tools

N/A

3.5. Impact on Data Warehouse

NA

3.6. Impact on Disaster Recovery

Proposed Change:

Revision: 1

Function: Build

Product Owner:

Developers : WSS Consultant

Environment : PROD

Date :

6.0. Installation process steps

6.1. Unix Admin tier

#	During release to QA	During Production release	During DR release
1	Get the MARSBalance.java, MARSFile.java, MARSFile.properties and SQLStatements.java classes and make a backup of them	Get the MARSBalance.java, MARSFile.java, MARSFile.properties and SQLStatements.java classes and make a backup of them	Get the MARSBalance.java, MARSFile.java, MARSFile.properties and SQLStatements.java classes and make a backup of them
2	Copy the MARSBalance.java from Dev2: …/mars	Copy the MARSBalance.java from Dev2: …/mars	Copy the MARSBalance.java from Dev2: …/mars
3	QA Paste the MARSBalance.java gotten in the next path: …/mars	Production Paste the MARSBalance.java gotten in the next path: …/mars	Production Paste the MARSBalance.java gotten in the next path: …/mars
4	Copy the SQLStatements.java from Dev2: …/mars	Copy the SQLStatements.java from Dev2: …/mars	Copy the SQLStatements.java from Dev2: …/mars

5	QA Paste the SQLStatements.java gotten in the next path: …/mars	Production Paste the SQLStatements.java gotten in the next path: …/mars	Production Paste the SQLStatements.java gotten in the next path: …/mars
6	Copy the MARSFile.java from Dev2: …/mars	Copy the MARSFile.java from Dev2: …/mars	Copy the MARSFile.java from Dev2: …/mars
7	QA Paste the MARSFile.java gotten in the next path: …/mars	Production Paste the MARSFile.java gotten in the next path: …/mars	Production Paste the MARSFile.java gotten in the next path: …/mars
8	Open MARSFile.properties which is in …/mars Update next line info.fileCreated=\n* rows reported. \nFile created correctly. By info.fileCreated=\n* . \nFile created correctly. Save the file.	Open MARSFile.properties which is in …/mars Update next line info.fileCreated=\n* rows reported. \nFile created correctly. By info.fileCreated=\n* . \nFile created correctly. Save the file.	Open MARSFile.properties which is in …/mars Update next line info.fileCreated=\n* rows reported. \nFile created correctly. By info.fileCreated=\n* . \nFile created correctly. Save the file.

6.2. Special notes

- N/A

7.0 Rollback procedures

7.1 Unix Admin tier rollback steps

#	During release to QA1 TRSTWPRCTXAPP03	During Production release trstsprwssapv01	During DR release TRALSDRWSSAPV01
1	Get the MARSBalance.java, MARSFile.java, MARSFile.properties and SQLStatements.java backup.	Get the MARSBalance.java and SQLStatements.java classes backup.	Get the MARSBalance.java and SQLStatements.java classes backup.
2	Copy the MARSBalance.java backup	Copy the MARSBalance.java backup	Copy the MARSBalance.java backup
3	Paste the MARSBalance.java gotten in the next path: …/mars	Paste the MARSBalance.java gotten in the next path: …/mars	Paste the MARSBalance.java gotten in the next path: …/mars
4	Copy the SQLStatements.java:	Copy the SQLStatements.java:	Copy the SQLStatements.java:
5	Paste the SQLStatements.java gotten in the next path: …/mars	Paste the SQLStatements.java gotten in the next path: …/mars	Paste the SQLStatements.java gotten in the next path: …/mars
6	Copy the MARSFile.java	Copy the MARSFile.java	Copy the MARSFile.java
7	Paste the SQLStatements.java gotten in the next path: …/ge/	Paste the SQLStatements.java gotten in the next path: …/ge/	Paste the SQLStatements.java gotten in the next path: …/ge/
8	Copy the MARSFile.properties	Copy the MARSFile.properties	Copy the MARSFile.properties
9	Paste the SQLStatements.java gotten in the next path: …/ge/	Paste the MARSFile.properties gotten in the next path: …/ge/	Paste the MARSFile.properties gotten in the next path: …/ge/

8.0 Additional information required by object type

8.1. New Package
8.2. New Table
8.3. New Database
8.4. New Scheduled Task

5.0 Master List of Objects to be installed

Estimated Install time: 45 mins.

Server: ICLPROD

Database: ICL

Other:

Resultados

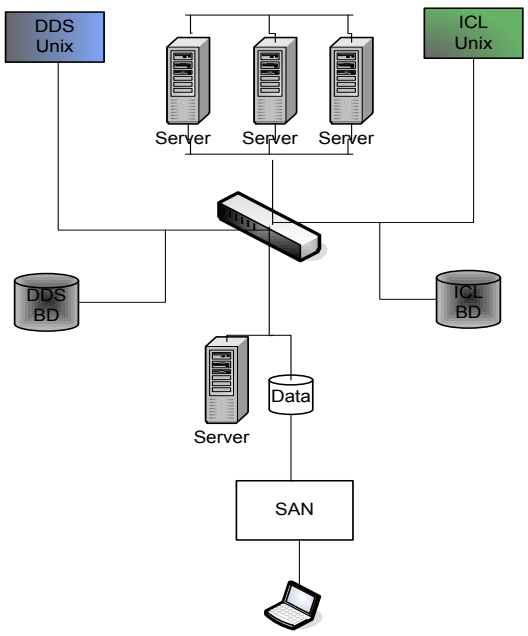
Se corren pruebas en Desarrollo con éxito
Se corren pruebas en Calidad con éxito
Se corren pruebas en Producción con éxito

Diseño de script para matar los procesos hijo de onyx

Problema:
Durante el reseteo del cluster de la aplicación DDS los procesos hijo de ONYX se quedan colgados
Proceso para su solución:

1. Hacer un análisis de la arquitectura del cluster
2. Identificar scripts encargados del reseteo.
3. Modificar los scripts encargados del reseteo
4. Correr pruebas en el ambiente de desarrollo.
5. Si las pruebas son exitosas en el ambiente de desarrollo, hacer la implementación en Calidad y correr pruebas.
6. Si las pruebas son exitosas en el ambiente de Calidad, hacer la implementación en Producción y correr pruebas.

Análisis de Arquitectura:



El cluster contiene DDS e ICL ambos con sus bases de datos, la SAN y este es accedido por Citrix.
Una red SAN se distingue de otros modos de almacenamiento en red por el modo de acceso a bajo nivel. El tipo de tráfico en una SAN es muy similar al de los discos duros como ATA,SATA y SCSI. En otros métodos de almacenamiento, (como SMB o NFS), el

servidor solicita un determinado fichero. La mayoría de las SAN actuales usan el protocolo SCSI para acceder a los datos de la SAN, aunque no usen interfaces físicas SCSI. Este tipo de redes de datos se han utilizado y se utilizan tradicionalmente en grandes main frames como en IBM, SUN o HP. Aunque recientemente con la incorporación de Microsoft se ha empezado a utilizar en máquinas con sistemas operativos Microsoft.
Una SAN es una red de almacenamiento dedicada que proporciona acceso de nivel de bloque a LUNs. Un LUN, o número de unidad lógica, es un disco virtual proporcionado por la SAN. El administrador del sistema tiene el mismo acceso y los derechos a la LUN como si fuera un disco directamente conectado a la misma. El administrador puede particionar y formatear el disco en cualquier medio que él elija.

Identificación de Scripts
El administrador baja el cluster y el script encargado de levantar el proceso onyx es start_dds_trm_onyx.ksh

Se procede al diseño del Script start_dds_trm_onyx.ksh

Pasos para ejecutar las pruebas en desarrollo.
Pasos para ejecutar las pruebas de Onyx en DDS

paso 0
ejecutar grep para obtener los PIDs:
/usr/ucb/ps -auxwww|grep onyx
paso 1
- Obtén los dos PIDs de los procesos que están corriendo en ese momento.
- Verifica el archivo :
…/pid/onyx_basics.pid
- En ese archivo agrega el PID que falte

paso 2
Ya agregado el PID en el archivo onyx_basics.pid, da de baja desde cluster el proceso ONYX.
paso 3
Verificar que ya no estén corriendo los dos procesos ONYX en el server.
paso 4

Respaldar el archivo :
…/start_dds_trm_onyx.ksh
paso 5
Copiar el script modificado como se especifica abajo:
fuente: …/start_dds_trm_onyx.ksh
Destino: …/start_dds_trm_onyx.ksh
paso 6
Levantar el servicio ONYX desde cluster y verificar el archivo contenga los 2 pids
el path es …/onyx_basics.pid
NOTA * si el archivo …/onyx_basics.pid no contiene los 2 PID's favor de agregarlos tal como se describe en el paso 1

paso 7

Bajar el proceso ONYX desde el cluster y verificar que no estén corriendo esos procesos (PID's del paso 1).

paso 8

Después de las pruebas hay que regresar el archivo start_dds_trm_onyx.ksh a su estado original. (respaldo del paso 4)

paso 9

Después de las pruebas proceder normalmente con la actividad del reinicio del server.

Paso 10

Explicación de los resultados esperados:

El Script modificado en el paso 5 va a dar de alta de forma automática los PID's hijo del proceso ONYX, en el paso 6 hay que verificar que efectivamente Estos PID's fueron adicionados en el archivo …/pid/onyx_basics.pid, en el paso 7 al bajar el proceso ONYX se espera garantizar que tanto el PID padre como los PID hijo del proceso ONYX mueran por ende se espera lo siguiente:

Si ningún proceso ésta corriendo, la prueba fué exitosa

Si un proceso quedo arriba, la prueba fallo

Enviar resultados y comentarios a:

Sergio Boullosa, Armando Rodríguez y Ernesto Pérez

Resultados Obtenidos del equipo de Unix

Hice las pruebas con DDS Onyx y no fueron satisfactorias. El archivo PID fué generado pero sin contenido, para corregir ésta situación hay que investigar mas a fondo que es lo que ésta pasando. Por lo pronto deje el script como estaba inicialmente.

Acciones.

Debido a que las pruebas fallaron se procede se hizo un análisis del proceso de reseteo y se toma la decisión de modificar el script init.sh

Pasos para ejecutar las pruebas de Onyx en DDS

paso 0

respaldar el script init.sh

paso 1

implementar el nuevo script init.sh

paso 2

dar de baja desde cluster el proceso ONYX.

paso 3

Levantar el servicio ONYX desde cluster

paso 4

Verificar los Pids agregados en el archivo

paso 5

Una vez que se verifiquen los pids hacer rollback y proceder normalmente con la actividad del reinicio del server.

Resultados esperados:

El Script modificado init.sh va a dar de alta de forma automática los PID's hijo del proceso ONYX hay que verificar que efectivamente estos PID's fueron adicionados en el archivo …/onyx_basics.pid, al bajar el proceso ONYX se espera garantizar que tanto el PID padre como los PID hijo del proceso ONYX mueran por ende se espera lo siguiente:

Si ningún proceso ésta corriendo, la prueba fué exitosa
Si un proceso quedo arriba, la prueba fallo
Enviar resultados y comentarios a:
Sergio Boullosa, Armando Rodríguez y Ernesto Pérez

Resultados Obtenidos del equipo de Unix

Ya hice las pruebas en DDS con el nuevo script init.sh. Teniendo como resultado "Satisfactorio" en todas las pruebas realizadas. Una vez terminadas las pruebas deje el script init.sh como estaba antes de las mismas.

Diseño de SOP para capturar todas las excepciones de la actividad TRM-TO-CARS

Problema:
Se requiere capturar todas las excepciones antes de ejecutar la actividad TRM-TO-CARS
Proceso para su solución:

1. Hacer un análisis de la arquitectura de la actividad.
2. Generar un SOP
3. Correr pruebas en el ambiente de desarrollo.
4. Si las pruebas son exitosas en el ambiente de desarrollo, hacer la implementación en Calidad y correr pruebas.
5. Si las pruebas son exitosas en el ambiente de Calidad, hacer la implementación en Producción y correr pruebas.

Generación del SOP

When to execute

CARS EXCEPTION and SELECTION script.
2nd Fiscal Sunday of Every Month.
Last Fiscal Sunday of Every Month.
On User request

Prerequisites

- DBO user access to WSS-ICF database.

SOP Steps to Execute

Steps to execute,
All the steps apply for
DEV
QA

DR
Production

Steps for CARS script

1. Log to ICF database

Log on to ICF database using SQL Developer with the DBO user.

2. Run Cars Script

Open the following script using sql developer by double clicking the next icon and accepting the prompt message. This script extracts the detailed feed error before executing the activity.

CARS_DETAILED_FEED_ERROR.txt

- Locate line 74 in the script and change the value of string '11/15/10' to the needed due date using the format 'MM/DD/YYYY'.
- For example
  - December 2003 Line 74 should be: '12/15/2003'
  - November 2004 Line 74 should be: '11/15/2004'
  - November 2008 Line 74 should be: '11/15/2008'
  - If we need to run for a period that is a future date (i.e. December 2011 use the current month) If the current month is September Line 74 should be: '09/15/2010';
- This query has another parameter that you can change, it is the ledger id, by default is 'GE' and is located at line 70.
- Run the script by clicking the green 'play' button or toggling the F9 key (this will take some minutes to finish).
- Right click in the result data under SQL editing area.
- Choose 'Export Data' → CSV or XLS
- Name the file as DATA.EXCEPTION.TREMA_mmddyy

3. Count the CARS Selections and Exceptions.

- After running the query run the following selects to check if there is CARS Entry data to select by executing next script.
- Keep in mind to change the due date and ledger group (if applies) located at 83 and 79 lines respectively.

CARS_COUNT_SELECTIONS.txt

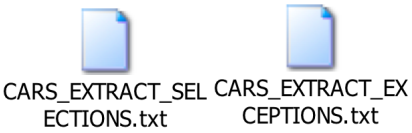
- This scripts uses DTS validations to extract selections or exceptions data, also, validations applies to next scripts, so they use regex data processing based on **ICL TREMA FILES FOR CARS 442** Document, if you have any questions over selection criteria, please refer this document before.
- To retrieve the exception count list, please execute the following script

- As the last query, also change the due date and ledger group (if applies) located at 83 and 79 lines respectively.

- If any of the selects throws records continue with step 4, else repeat step 2 for other dates.
- Use the following template to store the records count (this is an example when the requested date is from Jan 2003 to July 2010)

4. Extract the CARS Selections and Exceptions.

- Run the following selects after change their due dates and ledger groups (if apply) located on 83 and 79 lines respectively for CARS_EXTRACT_SELECTIONS, 110 & 106 for CARS_EXTRACT_EXCEPTIONS.



- Right click in the result data.
- Choose 'Export Data' → CSV or XLS
- Name the file as CARSENTRYSELECTIONS_mmddyy or CARSENTRYEXCEPTIONS_mmddyy as corresponds.

Please note that CARS_EXTRACT_EXCEPTIONS will help you to correct the exception values by aiming the incorrect field data, you can find and annotation like **"## Null value ##"** if the field has a null value on a row, or **'## WRONG ##** ' if the expected data doesn't match the expected value.

5. Save and store the file

1. Open the exported files and rename their "Export Worksheet" sheet according to the selected period mmddyy please be aware of use only English words. You may delete, in case of CSV export file, the second sheet that contains the SQL instructions.

2. Place the extract files in the following Citrix Path L:\ICF vs. MARS CARS\CARS Exception Reports. (It is the same path for Prod, DR, QA and Dev Citrix Environments)
3. Create the current year folder if doesn't exists, if exist go into it.
4. For each execution, a new sub-folder should be created with the name "CARS_Exception_mm-dd-yy", and it has to be placed in the current year folder.
5. The data obtained for each execution should be placed in the sub-folder created.

Expected Results

- CARS Selections and Exceptions delivered for requested dates.

Expected Duration CARS report

For each period could take from 5 to10 minutes. There are several factors to be consider, this is considered for CARS script.

Required Approvals

None.

Final Verification / Notification Immediately

- Send an email with the path of the files as example "L:\....\CARS Exception Reports".

- Verify if there were records in tables TMPCARSENTRYDET and TMPCARSENTRYEXCEPTION, if so; send the template to the requestor and to the contact people the data obtained. This is for CARS script

Expected Output

1. CARS Exception/Selection Reports for selected time criteria

Contact

Send Summarized output & exact location of files to:
DL: @....Treasury Trema Support Team
@...Softtek WSS Support

In Case of SOP Failures

Escalate this issue to as per the escalation guidelines.

○

Resultados de las pruebas
Los resultados obtenidos en Desarrollo fueron exitosos
Los resultados obtenidos en Calidad fueron exitosos
Los resultados obtenidos en Producción fueron exitosos

Bibliografía

- Documentación oficial de la Sun Microsystems de Java Development Kit Version 1.5

- Manual QUE - Special Edition Using Java, 2nd Edition, versión encontrada en internet (en inglés).

- Java2 Todo&Más, J. Jaworski, SAMS Publishing - APOGEO (en inglés)

- Javatm 2D Graphics, J. Knudsen, O'REILLY (en inglés)