

Ronja – Una plataforma de aplicación Java

Magnus Karlson

Ronja es un marco general de desarrollo Java basado en la TSP que permite el desarrollo rápido de, y el costo de mantenimiento reducido para, servidores en la nueva infraestructura de telecomunicaciones. El objetivo principal de diseño del marco general ha sido de usar productos standard, en el acto, para facilitar la adquisición de componentes externos y para sacar ventaja de las economías de escala.

El marco general Ronja incluye una biblioteca de programas de abstracción de alta disponibilidad y alto nivel y alta disponibilidad, lo que significa que los programadores no necesitan atascarse con los problemas de bajo nivel de crear estos programas – otro beneficio que se traduce en un tiempo de desarrollo más corto.

El autor describe los requisitos para aplicaciones adaptables y de alta disponibilidad, la arquitectura de hardware, la vista de Ronja de ambientes de hardware y ejecución, la estructura de una aplicación Ronja, y servicios de marco general.

Una arquitectura basada en componentes

Ronja es uno de los componentes en la plataforma recientemente sacada al mercado para el desarrollo de servidores de transacciones centrales y de control en la infraestructura de telecomunicaciones. Esta plataforma, llamada TSP, contiene un juego grande de componentes, que incluyen componentes de operación y mantenimiento (O&M), sistemas de operación, software personalizado / marcos generales, pilas de señales, y hardware. Estos componentes, debido a que tienen interfaces estandarizados bien definidos, pueden ser combinados en muchas maneras distintas para formar plataformas apropiadas para distintos servidores en redes de telecomunicaciones. La idea de definir una arquitectura basada en componentes con interfaces estandarizados es de asegurar que la arquitectura ha de prevalecer en el futuro al hacer que sea fácil intercambiar los componentes si se llegara a encontrar una mejor solución (Figura 1).

El primer producto que usa Ronja es el nuevo servidor de red de radio (RNS).¹

Objetivos de diseño

Ronja es un software personalizado / un marco general que ha sido desarrollado para ofrecer características de telecomunicaciones standard, tales como:

- alta disponibilidad – el software personalizado debe poder gestionar sistemas de multiprocesadores de acoplamiento flojo en los cuales la tolerancia de fallos y la fiabilidad se basan en redundancia $n + 1$ de elementos de procesamiento – o sea que el software debe poder gestionar transpaso, failover, etc. (con

failover queremos decir la funcionalidad para volver a poner en funcionamiento una unidad de aplicación que ha fallado).

- adaptabilidad – el software personalizado debe poder apoyar la distribución de una aplicación por procesadores múltiples; y
 - sustitución de código activo – el software debe permitir que todas las partes del sistema, tanto el hardware como el software (inclusive los sistemas de software personalizado y de operación), sean puestas al día mientras que el sistema se encuentra en funcionamiento, y con una degradación mínima de comportamiento.
- Para reducir a un mínimo los costos de desarrollo así como el costo de ser propietario de aplicaciones, se diseñó a Ronja con objetivos de diseño muy precisos, tales como
- hacer fácil reusar tecnología;
 - usar sistemas standard de procesadores y de operación
 - para permitir un alto nivel de abastecimiento externo;
 - para llevar a cabo y desarrollar sistemas que tienen las mismas condiciones técnicas;
 - hacer la funcionalidad de alta disponibilidad y de aglomeración lo más transparente posible para las aplicaciones;
 - apoyar varios lenguajes de programación;
 - facilitar la carga de software; y
 - permitir el uso de herramientas y entornos de desarrollo de software y pruebas de corriente principal.

El tratar de alcanzar estos objetivos es el motivo principal de elegir Java como el lenguaje principal de implementación. El diseño de funcionalidad de alto nivel, transparente de alta disponibilidad / aglomeración exige las altas capas de abstracción y protección proporcionadas por Java. La facilidad de cargar es inherente en Java, y herramientas nuevas de desarrollo y pruebas se encuentran disponibles también en una fase muy temprana para un lenguaje tal como Java.

La implementación actual de Ronja está basada en la alta disponibilidad y el software de aglomeración que se usa en la cartera de productos de servicios generales de radio por paquetes (GPRS). Sin embargo, la idea es de asegurar que cualquier marco general de este tipo pueda ser usado sin afectar futuros programas de aplicación. Hay que tomar nota de que la elección de un entorno de alta disponibilidad y aglomeración define el nivel de supervisión que se puede aplicar a las partes que no han sido desarrolladas directamente para la plataforma. A pesar de que Ronja define métodos de comunicar con programas o partes de programas escritos en otros lenguajes, el marco general subyacente debe dar apoyo para poner en marcha, parar, hacer adiciones y supervisar si las partes han de mostrar características de telecomunicaciones plenas para la aplicación completa.

CUADRO A, TERMINOS Y ACRONIMOS

3GPP	Third-generation Partnership Project
API	Application program interface
CI	Capsule instance
GEM	Generic Ericsson magazine
GPRS	General packet radio service
IRP	Integration reference point
JVM	Java virtual machine
O&M	Operation and maintenance
PI	Processor instance
RNS	Radio network server
Thread	Proceso de muy poco peso

Requisitos para aplicaciones adaptables de alta disponibilidad

Desafortunadamente no hay ninguna manera mágica para que algún software personalizado pueda crear una aplicación adaptable de alta disponibilidad de un programa ordinario escrito en lenguajes tradicionales (C, C++, Java, etc.) a no ser que el programa sea diseñado desde el principio para cumplir con algunos principios de diseño bastante obvios:

- los algoritmos que se usan deben estar distribuidos;
- varias instancias de la parte de la aplicación que lleva a cabo el trabajo deben poder coexistir, o trabajar en paralelo, o ambas cosas; y
- si las partes de programa han de ser libres de fallos y se les pueda hacer adiciones mientras se encuentran en servicio, debe ser posible también de repetir o conservar la información de estado que se necesita para que una nueva versión pueda recoger la información y continuar después de que se ha efectuado un traspaso o una conmutación.

Estas funciones pueden estar más o menos ocultas en la plataforma. El propósito de Ronja es al fin y al cabo de hacer ésto lo más simple que sea posible para el programador.

Arquitectura de hardware

Nuestro enfoque para resolver los problemas de alta disponibilidad (adaptabilidad, fiabilidad, y hacer adiciones en servicio) es de montar sistemas de multiprocesadores que no tienen ningún punto de fallo. Esto significa en la práctica que todos los componentes, procesadores, redes, interfaces de red, fuentes de energía, y ventiladores de enfriamiento están duplicados por lo menos o les es posible explotar la redundancia $n + 1$. En TSP usamos una nueva arquitectura de hardware que se llama el almacén genérico Ericsson (GEM). En un sistema que se ha instalado usando usando Ronja, exige ésto un almacén (repisa) que va provisto de tarjetas procesadoras basadas en SPARC, conmutadores Ethernet y puentes o encaminadores junto con cualquier hardware especial que podría necesitar un nodo de servidor específico (Figura 2).

La vista de Ronja de los entornos de hardware y ejecución

Una aglomeración contiene varias instancias de procesador (PI – los procesadores disponibles en la aglomeración) que pueden ser usadas por una aplicación. Cualquier número de instancias de cápsula (que es un término agregado para un entorno de procesamiento) puede ser puesto en

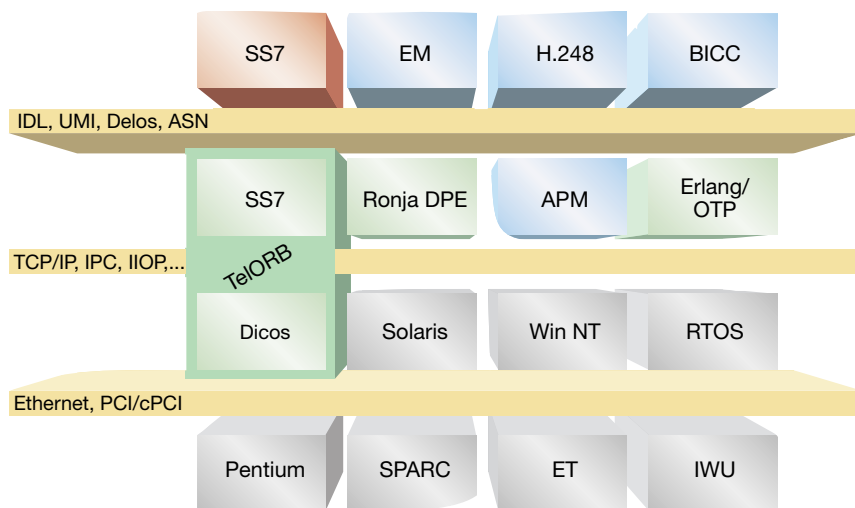
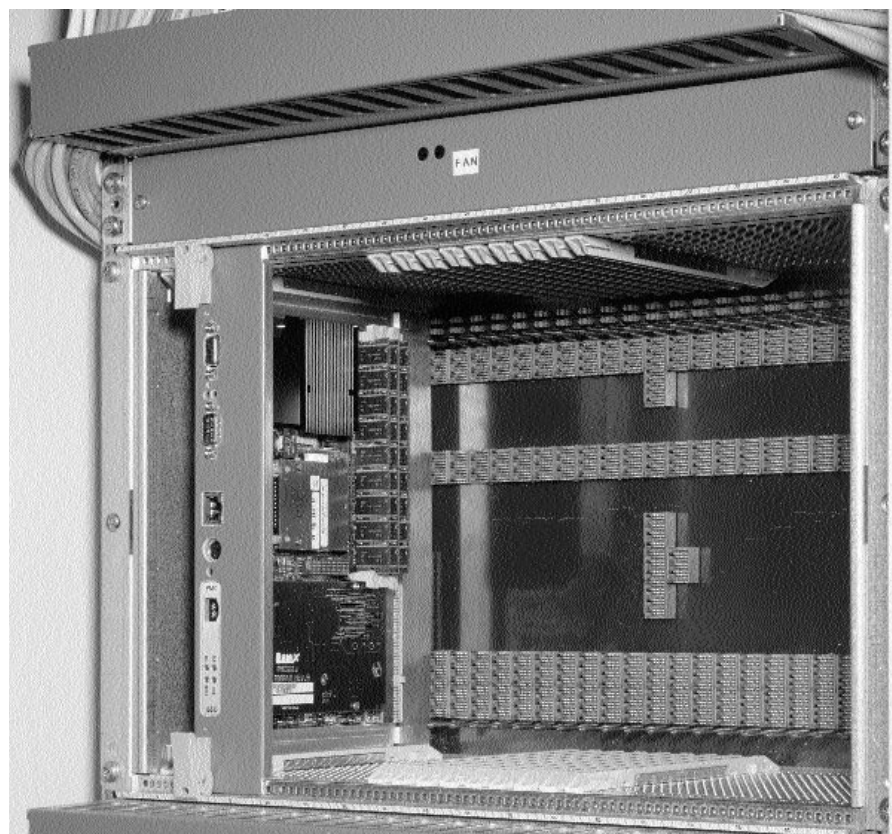


Figura 1
Arquitectura de componentes de la TSP.

Figura 2
Repisa típica de hardware para Ronja.



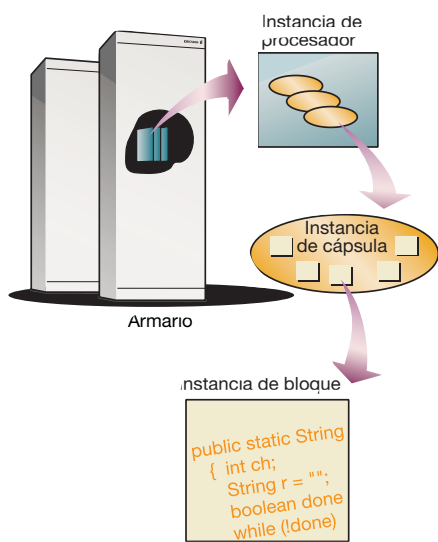


Figura 3
Entorno de ejecución.

marcha en estas instancias de procesador. En el marco general Ronja corresponde una cápsula normalmente a una máquina virtual Java (JVM) para ejecución Java. Para otros lenguajes, una cápsula es normalmente el equivalente de un programa (C, C++) ó donde sea apropiado, una máquina virtual (tal como Perl ó Erlang). Las aplicaciones son traducidas después a estas instancias de cápsula (Figura 3).

Estructura de una aplicación Ronja

Una aplicación Ronja está dividida en partes más pequeñas que se llaman bloques, que son las entidades más pequeñas que pueden ser mantenidas por el marco general. Por una “entidad mantenida” queremos decir una que puede ser cargada, puesta en marcha, parada, supervisada, que se le pueden hacer adiciones y gestionar versiones. Un bloque puede ser escrito en cualquier idioma, lo que significa que una aplicación puede consistir de bloques escritos en distintos lenguajes. Un bloque (el bloque de la raíz) tiene la responsabilidad de la comunicación con el marco general y de definir o supervisar la estructura de distribución y las funciones de la aplicación. Debido a que Ronja es una plataforma Java, se debe escribir el bloque de la raíz en Java para sacar ventaja completa de las características que Ronja puede proporcionar (Figura 4).

El papel del bloque de la raíz

La biblioteca de aglomeración y las plantillas de alto nivel y alta disponibilidad llegan a ser muy útiles en el bloque de la raíz. La mayor parte de las funciones de gestión de aplicaciones son manejadas de una forma más o menos transparente en la plantilla por medio de perfiles normalizados que el programador de aplicaciones usa para definir cómo se comportará la aplicación. Cuando se está definiendo el bloque de la raíz, puede elegir el programador entre perfiles que son suministrados por Ronja o desarrollados por el proyecto para gestionar la distribución, el hacer adiciones, la migración y el volver a poner en marcha la aplicación o partes de la aplicación. Cuando se pone en marcha una aplicación en Ronja, es en realidad el bloque de la raíz que se pone en marcha. El bloque de la raíz pone en marcha después el resto de la aplicación al recurrir a los perfiles apropiados de manera tal que se correlacione la aplicación a las instancias de

procesador disponibles y a distintas instancias de cápsulas. El bloque de la raíz juega después de esto principalmente el papel de un supervisor. Se le informa de cualquier cambio que afecta a la aplicación mientras que se encuentra en funcionamiento – tal como la adición o la remoción de hardware nuevo, o el fallo de algún bloque que forma parte de la aplicación, o peticiones para actualizar el software – y toma la medida apropiada usando los perfiles que corresponden para remediar la situación o cumplir con peticiones. Las plantillas y los perfiles definidos por Ronja son suficientes para casi todas las aplicaciones; sólo en casos muy raros o especiales debe ser necesario que el programador escriba perfiles únicos.

El bloque de aplicaciones

Ronja pone sólo pocos requisitos en la implementación de bloques de aplicaciones “normales”. El bloque necesita fundamentalmente sólo tres interfaces para poner en marcha, parar, y recibir información o datos. Un bloque de plantilla en Ronja ayuda al programador a diseñar el programa.

Abastecimiento de componentes externos

La facilidad de abastecimiento fue un objetivo principal cuando Ronja fue diseñada. Hay varios motivos para esto:

- Ericsson ya tiene muchos componentes de software que se pueden reutilizar;
- hay compañías especializadas que producen cantidades importantes de software útil; y
- mucho del software standard de facto usado en las aplicaciones Internet, por ejemplo, no se produce por cuenta propia en Ericsson.

Para sacar ventaja del software adquirido debe haber la menor cantidad posible de restricciones y reglas en el marco general. Para programas “únicos”, tales como un servidor Web, un agente facilitador de objetos, etc., se puede usar frecuentemente el concepto de aplicación de alta disponibilidad del marco general para subir el nivel de supervisión (volver a poner en funcionamiento si ocurre un fallo). Para incluir una detección de error o funcionalidad para volver a poner en funcionamiento una unidad de aplicación que ha fallado más sofisticada, es necesario escribir pequeños programas de supervisión que pueden poner en marcha la aplicación adquirida. Sin embargo nosotros necesitamos frecuen-

MARCAS REGISTRADAS

Todas las marcas registradas SPARC son usadas bajo licencia y son marcas registradas o marcas de fábrica de SPARC International, Inc. en los Estados Unidos y otros países. Los productos que llevan las marcas registradas SPARC están basados en una arquitectura desarrollada por Sun Microsystems, Inc.

temente tecnología o software a un nivel más integrado, tal como partes de un programa (pilas de comunicación, bibliotecas de conversión, simples agentes de protocolo de gestión de red). El marco general no debe poner restricciones en estos casos al tipo de llamadas de sistema o construcción de lenguaje que podría ser usado. Ronja no impone ninguna restricción de este tipo. Si el software carece de algunas de las funciones (puesta en marcha y parada, por ejemplo), puede ser que una plantilla de envoltorio que emula algunas de estas funciones pueda resolver fácilmente el problema. En algunos casos extremos – por ejemplo, si el componente adquirido no puede ser parado o está usando hebras (threads) de una manera no controlada – se puede hacer funcionar el componente adquirido en una cápsula separada (una cápsula se puede parar siempre de forma forzada) para que pueda ser parado o se le puedan hacer adiciones de una manera controlada.

Servicios de marco general

El marco general contiene también mucho de la funcionalidad genérica que necesitan las aplicaciones de telecomunicaciones. Los ejemplos típicos de este tipo de funcionalidad son los reguladores de tiempo, la designación de nombres, la comunicación, la información de estado (del sistema), canales de eventos, ficheros de conexión, y datos de configuración. Cada vez que se define la funcionalidad en alguna norma utilizable, cumple Ronja por lo general con la norma – en este contexto podemos mencionar el agente de servicio de alarma y notificación que cumple con IRP, que se encuentra disponible en la plataforma. IRP, o punto de referencia de integración, es una especificación 3GPP.

A un proyecto de desarrollo le es también posible añadir sus propios servicios especiales al marco general usando un interfaz de programa de aplicación de extensión de marco general definido (API). Por regla general es necesario esto sólo para una funcionalidad muy especial que no puede ser programada usando constructivos de programación más comunes, tales como funciones de biblioteca o módulos y comunicación entre procesos. Para funciones normales no es esta la manera recomendada de resolver problemas. Esto es porque a los servicios de marco general no se les puede hacer adiciones en movimiento rápido sin que se tenga que volver a poner en funcionamiento. Por contraste, a los bloques ordinarios se les pueden hacer adiciones

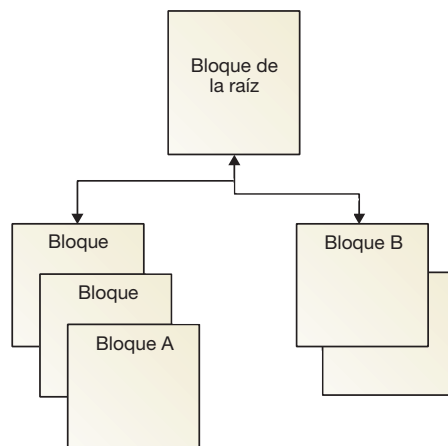


Figura 4
Estructura de aplicación.

mientras que una cápsula se encuentra en funcionamiento.

Conclusión

El marco general Ronja con software de alta disponibilidad / aglomeración entrega características de telecomunicaciones standard a aplicaciones que usan sistemas operacionales y hardware comerciales. La funcionalidad instantánea, la adaptabilidad, la ejecución sin interrupciones y la funcionalidad para volver a poner en funcionamiento una unidad de aplicación que ha fallado pueden ser implementadas de forma transparente en las aplicaciones usando la biblioteca de alta disponibilidad y alto nivel y las plantillas provistas en el marco general. Un juego ordinario de funciones y servicios de telecomunicaciones y de biblioteca de alta disponibilidad / aglomeración junto con las plantillas hacen posible una productividad muy alta cuando se están diseñando aplicaciones. El uso de sistemas operacionales y lenguajes standard permite que las personas que desarrollan y prueban puedan emplear las más recientes herramientas de corriente principal en su trabajo, asegurando de esta manera una alta productividad en fases posteriores de proyectos de desarrollo así como durante el mantenimiento. Ya que el marco general puede hacerse funcionar en estaciones de trabajo standard, no se necesita ningún hardware especial durante la mayor parte de las fases de desarrollo y pruebas. Esto reduce la necesidad de instalaciones especiales de hardware (caras y difíciles de mantener) durante el curso de proyectos.

REFERENCIAS

Musikka, N. y Rinnbäck, L.: El BSS basado en IP y el servidor de red de radio de Ericsson. Ericsson Review Vol. 77 (2000):4. pp. 224-233.