

Proyecto de investigación para la creación de modelos de negocio basados en software appliances y linux a medida

Resumen

Este documento describe un proyecto para construir un modelo de predicción de actualizaciones software. Los resultados de este proyecto permitirán crear modelos de negocio basados en software appliances y linux a medida.

I - Introducción

En los próximos años se prevé un importante incremento de la virtualización de sistemas operativos, debido a las tecnologías de virtualización tanto software (vmware [0], Xen [1], VirtualBox [2]) como hardware (tecnología VT de intel [3] y AMD V [4]), y, como consecuencia, también de la creación de distribuciones linux y *software appliances*. Una demostración de esta afirmación es que empresas tan importantes como Novell, RedHat, Intel, Amd, y Sun están invirtiendo en estas tecnologías, tanto de virtualización como de software appliances.

Las tecnologías de virtualización permiten que en un mismo equipo hardware puedan funcionar distintos sistemas operativos a la vez. Por ejemplo, en un servidor con virtualización podrían funcionar a la vez un sistema windows y un sistema linux. Esta tecnología está disponible para los procesadores x86 i x64, es decir, para los procesadores que encontramos en

los sistemas de escritorio, portátiles y servidores.

Gracias a las características de la virtualización, hoy en día nos encontramos que en un mismo servidor pueden estar funcionando distintas aplicaciones y cada una con sus librerías y sistema operativo. Por ejemplo, tenemos un servidor donde corren a la vez un sistema de monitorización, uno de detección de intrusiones, un gestor de correo y aplicaciones web echas a medida.

Estas configuraciones permiten un ahorro en la compra de equipos, pues podemos comprar un solo servidor y, a la vez, al instalar las distintas aplicaciones en él, un ahorro en los procesos de instalación y mantenimiento, pues como cada aplicación viene ya con su sistema operativo, los procesos de instalación y mantenimiento son más simples y por lo tanto menos costosos. Además, y también muy importante, permiten una independización mayor de los proveedores, con lo que esto significa en términos de competencia, creación de nuevas empresas, disminución de los precios, etc. debido a que cada proyecto nuevo es independiente del anterior, por lo que a sistemas se refiere, y por lo tanto puede encargarse a distintos proveedores de software.

Sin la virtualización, o se instala cada aplicación en un hardware distinto, o se instalan todas en un mismo sistema operativo, con los problemas de conflictos entre ellas, conflictos que son la causa principal de los costes de instalación y mantenimiento.

En este contexto, los software appliances ganan mucha importancia. Los software appliances son estas aplicaciones junto con su sistema operativo. Por ejemplo, un software appliance del CRM (software para la gestión de clientes) SugarCRM, sería el SugarCRM con las librerías y aplicaciones que necesita (servidor web, librerías php, mysql), más la configuración (configuración del php, del apache, del firewall, y de la base de datos), más un kernel de linux.

Las distribuciones linux son muy parecidas a los software appliances, es más, muchas veces se confunden ambos términos. En este proyecto, cuando hablamos de distribuciones linux, nos referimos a sistemas de carácter más general basados generalmente en otra distribución base como openSUSE o Ubuntu. Ejemplos de distribuciones linux las tenemos en las distribuciones autonómicas como son linex, guadalinux, max, molinux, linkat, y melinux. A diferencia de las distribuciones, los software appliances se basan en mini distribuciones, también denominadas JeOS, como por ejemplo LimeJeOS.

Debido al echo de que estas tecnologías son todavía muy nuevas, hay muy poca experiencia y no existen estudios sobre los costes de mantener tanto distribuciones linux como software appliances, que por otro lado es un factor

muy importante para la creación de modelos de negocio. Cuando se implementa una distribución linux de carácter general, como las distribuciones autonómicas, o se implementa un software appliance, como por ejemplo un software appliance para un CRM, el coste de dar soporte depende de la frecuencia en que salen nuevas actualizaciones de las distribuciones base. Cada vez que sale una actualización debe probarse y en algunos casos modificarse para adaptarla al nuevo sistema. Estas pruebas y modificaciones tienen un coste asociado en recursos, principalmente en personal altamente cualificado.

La frecuencia en que salen actualizaciones no esta predefinida, si no que las actualizaciones salen según se van encontrando errores. Esta frecuencia depende de muchos factores no observables como el número de personas que se dedican a revisar, el numero de personas que se dedican a implementar, y la madurez de los distintos componentes. La falta de estudios sobre la frecuencia de las actualizaciones en estos sistemas incurre en la incapacidad de crear modelos de negocio apropiados para esta nueva tecnología.

Para poder llevar a cabo los modelos de negocio, una pieza clave es poder predecir las actualizaciones. Con estas predicciones, las empresas podrían planear mejor la distribución de sus recursos, con las ventajas que ésto supondría en términos de costes y a la vez en la mejora de la calidad de los servicios ofrecidos y en productos más competitivos al pasar de un modelo de trabajo reactivo (respondiendo a las actualizaciones a medida que se van publicando y por lo tanto teniendo recursos libres disponibles) a uno más pro-activo (planeando con antelación la respuesta a las actualizaciones y por lo tanto planeando también los recursos disponibles). No obstante, y debido a su naturaleza, realizar estas predicciones no es trivial.

Este proyecto investigará y desarrollará un modelo predictivo de la publicación de actualizaciones de una distribución linux, para poder evaluar los costes de la creación de distribuciones nuevas y software appliances, y crear así una base para la creación de nuevos modelos de negocio y empresas españolas pioneras en esta nueva tecnología. Este modelo predictivo se publicará en forma de aplicación acompañado de artículos técnicos.

Organización de la propuesta

En la siguiente sección (Sección II) describimos la arquitectura del sistema que pretendemos desarrollar. Hemos diseñado el sistema como un sistema de procesamiento de flujo de datos (*stream system*). De forma parecida a los stream systems generales como STREAM [5], Borealis [6], SATware [7], y

TelegraphCQ [8], datos procedentes de sensores son continuamente manipulados e interpretados por operadores en su camino hacia los usuarios. De esta forma, nuestro sistema es modelado como un stream system donde los cyber-eventos [9] producidos por la publicación de actualizaciones son interpretados por una serie de operadores para producir las predicciones de las nuevas actualizaciones.

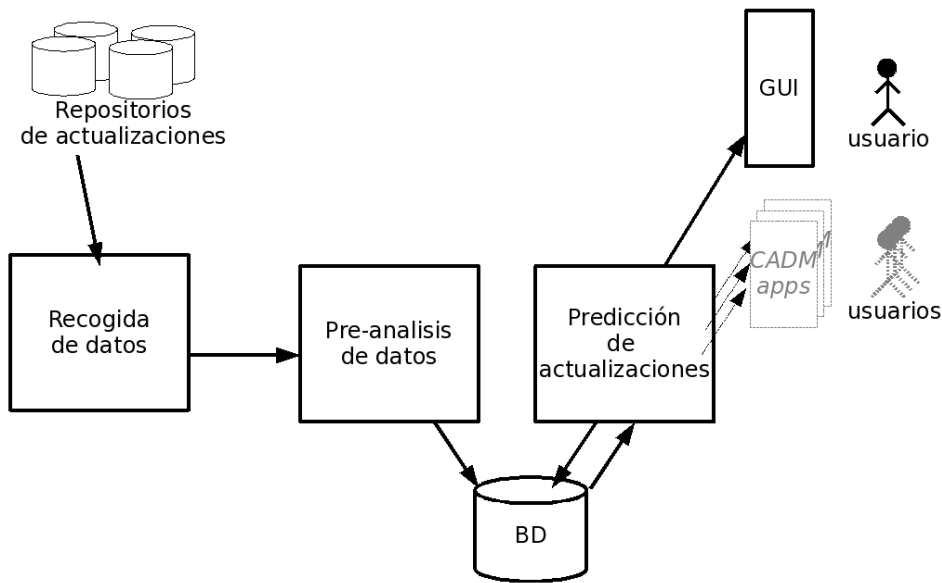
Este sistema contiene un módulo de predicción que genera, basado en la historia y según un modelo predictivo, una predicción futura. Este modelo se realimentará con los nuevos datos, procedentes del flujo de datos. El módulo de predicción es el problema central de investigación de este proyecto. La Sección III, además de especificar el funcionamiento esperado de dicho módulo, describe las características del problema que lo hacen no trivial y como pensamos enfocar el problema.

Después de la Sección III, la propuesta continua con la Sección IV donde explicamos que tecnologías vamos a utilizar para desarrollar el sistema. En la Sección V describimos los hitos necesarios para cumplir con el objetivo de la investigación, o dicho de otra forma, el plan de trabajo. La propuesta termina con una descripción de las personas involucradas en el proyecto (Sección VI) y concluye en la Sección VII.

II - Arquitectura del sistema

La arquitectura del sistema sigue el patrón típico de los sistemas de streaming. Los datos son obtenidos de una fuente externa (repositorios de actualizaciones), pasan por un módulo de interpretación/decodificación que extraerá datos estadísticos, y con éstos se obtendrá una predicción que será mostrada al usuario mediante una interfaz de usuario.

Los datos, como en todo sistema de streaming, serán recogidos de forma continua. Éstos datos realimentarán al sistema de predicción y, en consecuencia, la predicción y su visualización al usuario se irá actualizando en el tiempo. A continuación describimos el funcionamiento de cada una de los módulos principales (repositorios de actualizaciones, recogida de datos, pre-análisis de datos, predicción de actualizaciones, bases de datos, e interfaz de usuario).



Repositorios de actualizaciones: Es donde se producen los nuevos eventos, es decir, las nuevas actualizaciones. Estos repositorios pueden ser públicos o privados. En el primer caso suelen ser estructuras de directorios y ficheros en un servidor web. Según sea esta estructura y los ficheros que contenga, tenemos repositorios tipo yum, yast, you o apt. En el segundo, una aplicación web (como Novell Zenworks o RedHat Network).

Recogida de datos: El módulo de recogida de datos deberá conectarse a los repositorios de forma continua para comprobar si hay nuevas actualizaciones, y en el caso afirmativo, descargar los datos asociados a éstas.

Pre-análisis de datos: Una vez recolectados los datos, deberá extraerse información de éstos y realizar cálculos estadísticos necesarios para el módulo de predicción. Deberá también descartarse la información que no sea relevante.

Predicción de actualizaciones: El objetivo principal de la investigación. Con los datos estadísticos del módulo anterior, deberá generarse un modelo predictivo que nos permita, en el módulo de interfaz, presentar una predicción de las actualizaciones al usuario. Esta predicción se basará en los datos estadísticos de los datos recogidos hasta el momento, y en un modelo predictivo, que es el que hay que definir. Como en todo sistema de

streaming, los datos irán llegando de forma continua, por lo que el modelo debe ser capaz de realimentarse y recalcularse. Por otro lado, el usuario deberá especificar el conjunto de aplicaciones sobre las cuales quiere saber el modelo, además de la ventana de tiempo donde quiere calcular la predicción.

Bases de Datos (BD): Los datos estadísticos, así como los necesarios para las predicciones, se guardarán en BdD para que el sistema sea más robusto ante fallos o procesos de mantenimiento.

Interfaz de usuario: El módulo de interfaz de usuario será el que permita al usuario interactuar con el sistema. Aunque no es el objetivo del proyecto, la arquitectura contemplará la posibilidad de que la interfaz de usuario pueda ser reimplementada, ofreciendo las interficies de programación (APIs) correspondientes. No obstante, para la investigación se implementará una interfaz simplificada que permita al usuario entrar los parámetros, como son las aplicaciones de las cuales queremos la predicción y la ventana de tiempo, y obtener una representación gráfica del modelo de predicción.

III - Predicción

El modulo de predicción es el componente principal de investigación de este proyecto. Obtener un buen modelo de predicción es de vital importancia, pues los modelos de negocio dependen principalmente de éste. El modelo de predicción debe ser capaz de responder cuando sean las próximas actualizaciones. Concretamente, proponemos modelar el tiempo entre actualizaciones como una variable aleatoria (X) de la cual desconocemos su distribución de probabilidad. No solo desconocemos la distribución que sigue X (y los parámetros de dicha distribución como la media y la distribución estándar), sino que tampoco sabemos si siempre sigue la misma distribución o con los mismos parámetros. Peor aún, no sabemos si sigue ninguna distribución de probabilidad conocida.

Debido a la naturaleza de las actualizaciones, decidir cual es el mejor modelo para X no es trivial. Cuando será la próxima actualización depende de muchos factores. La mayoría de estos factores cambian con el tiempo y muchos de ellos no pueden ser observados directamente. Por ejemplo, cuando será la próxima actualización depende de parámetros que no són directamente observables y están sujetos a cambios como el número de programadores, la experiencia de los programadores, la dedicación de los programadores (a tiempo completo, durante su tiempo libre, etc), y la correcta planificación del proyecto (incluyendo los tests). Cuando será la próxima actualización también depende de factores observables y fijos como cuando fué la última actualización, el tipo de software (librería multimedia, driver de red, base de datos, etc), y el tipo de actualización (de seguridad, nuevas funcionalidades, etc). También depende de factores observables pero

dinámicos como la madurez del software e incluso puede depender de la publicación de actualizaciones de otro software. La relación de estos factores con la frecuencia de actualizaciones es, no obstante, desconocida; además, la mayoría de ellos no son directamente observables con lo cual no está claro como de precisas pueden ser las predicciones basadas en un modelo que solo tiene en cuenta los factores observables.

Para llevar a cabo las predicciones, el módulo de predicción seguirá la siguiente metodología. De forma periódica, el módulo de predicción intentará encontrar un modelo de distribución de probabilidad para X basado en cuando se publicaron las actualizaciones en el pasado. En general, este proceso se realizará de forma automática mediante la transformación de los datos y pruebas estadísticas estándares como el test Anderson-Darling, el Kolmogorov-Smirnov [19][16], el Pearson's chi-square [16], o similares. Aunque en general el proceso será automático, al principio será asistido por el investigador mediante la transformación y visualización de los datos y con la ayuda de las pruebas estadísticas estándares mencionadas. Debido al carácter dinámico de las actualizaciones, el módulo de predicción estimará continuamente (y con una frecuencia superior a la estimación de la distribución de probabilidad) los parámetros de la distribución como la media y la desviación estándar. Estos parámetros se podrán estimar con estadísticas descriptivas, algoritmos ML (maximum likelihood), o técnicas bayesianas [17][18]. En el caso de que no se pueda encontrar similitud con una distribución de probabilidades conocida, el modelo predictivo se basará en la estimación de parámetros como el rango, la media, la desviación estándar, y similares .

Además de dinámicamente construir un modelo de X , el modelo se enriquecerá con estudios sobre la relación entre las actualizaciones y los factores observables. Principalmente, el investigador estudiará la dependencia mediante pruebas estadísticas (como el test chi-square o coeficientes de correlación) de las actualizaciones con el tipo de software, el tipo de actualización, la madurez del software, y la publicación de actualizaciones de otras piezas de software.

IV - Implementación

Para la implementación de este proyecto, se utilizarán tecnologías habituales en las empresas de informática de nuestro país, así como estándares internacionales y prácticas habituales en proyectos opensource.

Se implementará básicamente en SUN J2EE [10] utilizando librerías como son por ejemplo struts [11], habituales en el sector y por tanto un estándar de facto en nuestra industria. La comunicación entre procesos se realizará mediante tecnologías de servicios web (SOAP [12] y XMLRPC [13]), estándar internacional de comunicación. Aunque pudiera implementarse

con otras tecnologías, como *Ruby on Rails*, la más extendida en nuestras empresas es J2EE, y por lo tanto la que nos puede mejor asegurar un futuro para el proyecto.

Por otro lado, todos los elementos utilizados serán de código abierto, según la interpretación de la Free Software Foundation [14], así como los implementados, de forma que pueda reaprovecharse el código en su totalidad por otras empresas o instituciones.

La metodología a utilizar será una metodología ágil [15], iterativa y evolutiva, con el objetivo de publicar a menudo y que cada publicación contenga pocas modificaciones, a la vez que el diseño vaya evolucionando hacia la arquitectura aquí propuesta, siguiendo así la tradición más aceptada en la comunidad *open source*.

Utilizando estas tecnologías y prácticas, se persigue un doble objetivo: por un lado, implementar lo mejor posible los requerimientos del proyecto, y por otro, que el proyecto, una vez terminado, pueda ser reutilizado y evolucionado por otras empresas de nuestro país.

V - El equipo

Daniel Massager (Investigador principal): 8 años de experiencia en I+D en laboratorios de la universidad en distintas universidades y países (España, Escocia, Estados Unidos) construyendo sistemas. Los dos últimos años ha estado en I+D de stream systems. Anteriormente estuvo involucrado en I+D de sistemas de simulación y gestión de redes utilizando técnicas de inteligencia artificial.

Pere Martinez (Director del proyecto): Director general y socio de Itechgrup. Dispone de amplia experiencia en el mercado empresarial TIC español. Anteriormente a Itechgrup estuvo 8 años en Novell donde participó en proyectos como el Mare Nostrum (Supercomputador de Barcelona), la distribución linux para Melilla Melinux, la distribución linux para el *Departament d'Educació de la Generalitat de Catalunya* Linkat, proyectos de implantación de linux en el ayuntamiento de Barcelona, el grupo Eroski, y muchos otros proyectos. Líder del grupo de trabajo de software libre y miembro de la junta directiva del *Clúster per a l'Innovació*, miembro asesor de CatPL (asociación de empresas de software libre en Cataluña), miembro asesor del COEIC (Colegio Oficial de Ingenieros en Informática de Cataluña), miembro del consejo rector de CETEI (Centro de Tecnología Ituarte), y participante en el club de innovación de Sitges y en ESLE (Empresas de Software Libre de Euskadi).

Jordi Massager (colaborador): experiencia en la creación de

distribuciones autonómicas (Linkat y Melinux) y software appliances (Openbravo-Network), 3 años de experiencia en empresas *open source* con tecnologías Java, Php, Python, linux y tecnologías de streaming de audio y video, miembro de openSUSE, certificado en SuSE linux Enterprise, colaborador habitual de la revista Todolinux, participante en el proyecto LimeJeOS-openSUSE, participante en el proyecto openSUSE-edu, creador del software instlux-openSUSE y 4 años de colaboración en proyectos de investigación en distintos grupos de investigación en España y Alemania.

V - Plan del proyecto

El proyecto es de un año y se divide en 4 trimestres. A continuación detallamos en qué nos vamos a concentrar en cada trimestre y qué sistemas e informes vamos a producir.

Primer trimestre: i) Implementar aplicación que genere un conjunto de estadísticas sobre las actualizaciones de la última versión de SUSE. ii) White paper sobre ello y publicación del código fuente.

Segundo trimestre: i) Evolución de esta aplicación para añadir más versiones de SUSE así como realimentarse. ii) Introducir un módulo predictivo básico. iii) White paper sobre ello y publicación del código fuente.

Tercer trimestre: i) Evolución del modelo predictivo. Aquí se aplicará la investigación en estadística y modelos predictivos. ii) White paper sobre ello y publicación del código fuente.

Cuarto trimestre: i) Evolución de la aplicación para que sea accesible desde web. Incluye separación de procesos y utilización de webservices para la comunicación entre éstos y diseño web. El diseño web no es el objetivo por lo que será un diseño simple. ii) White paper sobre la arquitectura del sistema y publicación del código fuente.

VI - Conclusiones

El objetivo final de este proyecto de investigación es crear una base para la creación de nuevos modelos de negocio para las empresas españolas basadas en una tecnología nueva, la de los software appliances, y por tanto ser pioneras en este sector. Para ello, debe desarrollarse un modelo predictivo de la publicación de actualizaciones de una distribución linux, para poder evaluar los costes de la creación de distribuciones nuevas y software appliances.

Este modelo predictivo se publicará en forma de aplicación acompañado de artículos técnicos. La aplicación se desarrollará en las tecnologías más usadas en las empresas españolas así como siguiendo estándares internacionales. Junto a este echo, su publicación como software libre, permitiendo a cualquier otra empresa española seguir con el desarrollo.

Bibliografía

- [0] VMware, "VMware: Virtualization via hypervisor, Virtual Machine, and Server Consolidation", <http://www.vmware.com>, 2008.
- [1] Citric Systems, "Xen", <http://www.xen.org>, 2005-2008.
- [2] Sun Microsystems, "VirtualBox", <http://www.virtualbox.org>, 2008.
- [3] Intel Corporation, "Intel Virtualization Technology (intel VT) in Computing", <http://www.intel.com/technology/virtualization/>, 2008.
- [4] Advanced Micro Devices (AMD), "Presentacion de AMD Virtualization", http://www.amd.com/es-es/Processors/ProductInformation/0,,30_118_8796_14287,00.html, 2008.
- [5] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, J. Widom, "STREAM: The Stanford Data Stream Management System", 2004.
- [6] Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Ugur Cetintemel, Mitch Cherniack, Jeong-Hyon Hwang, Wolfgang Lindner, Anurag S. Maskey, Alexander Rasin, Esther Ryzkina, Nesime Tatbul, Ying Xing, Stan Zdonik, "The Design of the Borealis Stream Processing Engine", CIDR 2005.
- [7] Bijit Hore, Hojjat Jafarpour, Ramesh Jain, Shengyue Ji, Daniel Massaguer, Sharad Mehrotra, Nalini Venkatasubramanian, Utz Westermann, "SATware: Middleware for Sentient Spaces, (Book chapter)", in "Multimodal Surveillance: Sensors, Algorithms and Systems", 2007.
- [8] Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy Samuel Madden, Frederick Reiss and Mehul A. Shah. "TelegraphCQ Continuous Dataflow Processing", SIGMOD 2003.
- [9] Ramesh Jain, "EventWeb: Developing a Human-Centered Computing System", IEEE Computer 41(2), pp. 42-50, 2008.
- [10] SUN Microsystems, "Java EE at a Glance", <http://java.sun.com/javaee>, 2008.
- [11] The Apache Software Foundation, "Struts", <http://struts.apache.org>, 2008.
- [12] Grupo de Trabajo sobre el Protocolo XML del W3C, "SOAP version 1.2, Recomendacion del W3C", <http://www.w3c.es/Traducciones/es/TR/2003/REC-soap12-part0-20030624/>, 2003-2008.
- [13] XMLRPC: <http://www.xmlrpc.com/>, 2008.
- [14] Free Software Foundation (FSF), <http://www.fsf.org>, 2004-2008.
- [15] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas, "Manifesto for Agile Software Development", <http://agilemanifesto.org/>, 2001, last accessed 2008.
- [16] Sarah Boslaugh, Paul Watters, "Statistics in a Nutshell: A Desktop Quick Reference", O'Reilly.
- [17] S. Russel, P. Norvig, "Artificial Intelligence: a modern approach-second edition", Prentice-Hall, 2003.
- [18] Alexander T. Ihler, Jon Hutchins, Padhraic Smyth, "Adaptive event detection with time-varying poisson processes", KDD 2006.
- [19] JUMK.de, "Statistic calculator", <http://jumk.de/statistic-calculator>, 2008.