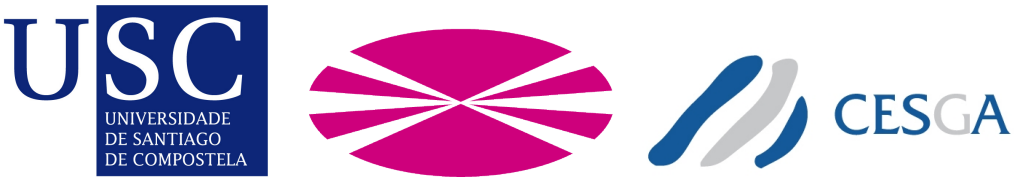

Mestrado en Computación de Altas Prestacións
Traballo de Fin de Mestrado

Deseño e implementación do despregamento na nube dunha plataforma altamente escalable de metadatos multimedia

Autor: JUAN FONT ALONSO
Directora: LAURA MILAGROS CASTRO SOUTO
Director: EMILIO JOSÉ PADRÓN GONZÁLEZ

A Coruña, a 5 de febreiro do 2015.

Resumo

Este proxecto busca explorar as potencialidades e vantaxes das plataformas de Infraestrutura como Servizo (IaaS) para o seu uso nun desenvolvemento con requerimentos de alta escalabilidade e dispoñibilidade. A posibilidade de modificar, sen deter a execución, a estrutura do clúster que proporciona o servizo constitúe un importantísimo aporte deste tipo de plataformas.

O traballo a desenvolver inclúe o deseño e implementación do despregamento nunha plataforma na nube dun servizo para proporcionar metadatos sobre contido multimedia para o seu uso en distintos dispositivos, buscando aproveitar as mencionadas posibilidades do modelo IaaS.

As tarefas realizadas enmárcanse dentro de PLATINO (PLATaforma Independiente Operativa para equipos de electrónica de consumo), un proxecto INNTERCONECTA de colaboración entre empresas e institutos tecnolóxicos, financiado por fondos FEDER, do Centro para o Desenvolvemento Tecnolóxico Industrial (CDTI) e da Consellería de Economía e Industria.

Palabras chave:

√ Erlang.

√ Python.

√ Amazon Web Services.

√ REST.

√ Infrastructure as a Service.

√ Escalabilidade.

Índice xeral

Página

Capítulo 1
Introdución

Índice xeral

No presente capítulo realizarase unha descrición do proxecto PLATINO, os seus obxectivos principais e a plataforma de computación na nube elixida para realizar o despregamento do compoñente principal de sistema, EPGcloud.

1.1. Proxecto PLATINO e EPGcloud

Nos últimos anos é perceptible un cambio nos patróns de ocio nos fogares; estase a ver como o centro de entretemento no fogar deixa de ser o ordenador e volve a ser a televisión. Esta tendencia, sen dúbida relacionada coa mellora dos equipamentos instalados nos fogares (televisores de gran tamaño, sistemas de audio multicanle, videoconsolas...), é vista por parte dos operadores de televisión coma unha oportunidade para incrementar e estender a oferta dos seus servizos, apostando polo vídeo baixo demanda e os servizos integrais para o fogar dixital.

Nesa liña preséntase PLATINO (PLATaforma Independiente Operativa para equipos de electrónica de consumo), un proxecto de desenvolvemento experimental con colaboración efectiva entre empresas e organismos públicos de investiga-

ción, enmarcado dentro da convocatoria FEDER INNTERCONECTA GALICIA (proxecto ITC-20113001). Este proxecto foi financiado polos fondos FEDER da Unión Europea (na partida "Fondo Tecnolóxico", un apartado especial dos fondos FEDER adicado á promoción do I+D+i empresarial en España), polo Centro para o Desenvolvemento Tecnolóxico Industrial (CDTI) e pola Consellería de Economía e Industria.

Os obxectivos do proxecto consisten no despregamento dunha experiencia de usuario unificada a través do desenvolvemento dunha plataforma operativa integral para ser incluída en todo tipo de dispositivos de electrónica de consumo, orientándose cara o concepto de Fogar Dixital Conectado. A idea subxacente tras esta iniciativa está en proporcionar entornos que non se vexan afectados polas limitacións dos dispositivos (móbiles, tabletas, set-top-boxes (STB), SmartTVs...) de xeito que o usuario interactue de maneira natural e transparente cos seus equipos.

O sistema EPGcloud é a parte do proxecto PLATINO encargada de proporcionar, a través dunha interface de programación, metadatos de contido multimedia que sirvan de base para enriquecer os medios audiovisuais. Un dos piares sobre os que se constrúe a plataforma son os servizos de guía electrónica de programas (EPG polas súas siglas en inglés). A partires de esa información de emisións poderán ser proporcionadas, por exemplo, unha serie de recomendacións personalizadas en base ao perfil do usuario. Búscase engadir valor ao contido, tanto ao usuario como á suministradora de dispositivos.

1.1.1. EPGcloud

Previa a introdución de EPGcloud, a obtención dos metadatos realizábase dun xeito semiautomático: varias veces ao día lanzábase un proceso por lotes que, mediante consultas á base de datos existente, xeraba unha serie de ficheiros de texto con toda a información necesaria. Estes datos eran logo proporcionados de xeito manual a certos clientes (por email, FTP...), ou servidos desde un servidor web. O modelo da base de datos existente, que continuará a ser empregada nos sistemas do operador cliente, está enfocado tamén a este xeito de traballar, con decisións de deseño difíciles de cambiar sen romper a compatibilidade con aplicacións xa despregadas.

Esta metodoloxía non era viable, por exemplo, en dispositivos móbiles, xa que

o proceso por lotes xera ficheiros de gran tamaño e reducida granularidade, pouco axeitados para estes dispositivos.

EPGcloud cambia totalmente esta aproximación. A información será proporcionada baixo demanda, a través dunha interface de programación (API, polas súas siglas en inglés) sinxela e coñecida, capaz de funcionar desde múltiples dispositivos. O usuario obterá sempre a os últimos datos dispoñibles, xa que a súa petición será realmente procesada. Ademais, será posible solicitar exactamente a información que se precisa, sen ter que descargar información extra, reducindo o consumo de largo de banda e de CPU.

EPGcloud traballará tamén sobre un novo deseño de base de datos, que corrixe as deficiencias e carencias encontradas no anterior modelo e que se adapta mellor ao xeito de traballar polo novo sistema.

Pola natureza dos dispositivos cliente da plataforma PLATINO e os patróns de consumo de contidos multimedia, é esperable recibir unha elevada carga de accesos simultáneos, cunha grande variabilidade segundo a franxa horaria. Polo tanto, conceptos como a escalabilidade e o alto rendemento definen parte dos obxectivos principais deste deseño. Polo mesmo motivo, a alta dispoñibilidade do sistema é esencial e, consecuentemente, as técnicas de tolerancia a fallos deben ser aplicadas ao longo do deseño e desenvolvemento.

Á xa mencionada variabilidade da carga ao longo do día haberá que engadir o número de usuarios previstos da plataforma, máis de 100.000.

Nun contexto como o descrito, un despregamento tradicional carece da flexibilidade necesaria; unha solución *cloud*, capaz de adaptar a súa capacidade en función da demanda prevista ou substituír de xeito inmediato un nodo que deixou de funcionar, presenta vantaxes dificilmente igualables.

1.1.2. Elementos de EPGcloud

O fin último de EPGcloud é presentar unha interface de programación para que desenvolvedores de aplicacións de distintas plataformas poidan enriquecer contido audiovisual en emisión, xa sexa televisión (terreste, cable, satélite ou televisión Over-The-Top) ou vídeo baixo demanda (VoD [18], polas súas siglas en inglés). Ademais das tarefas relacionadas coa obtención de datos en si mesma, unha plataforma deste tipo require servizos auxiliares coma autenticación de usuarios, mantemento da base de datos, facturación... que deben ser planificados,

desenvolvidos e despregados.

EPGcloud está formado por distintos módulos e subsistemas, traballando de xeito coordinado, sen que todos eles estean despregados nunha plataforma *cloud*. Para comunicarse entre eles empregan servizos REST [17] (ver Figura 1.1).

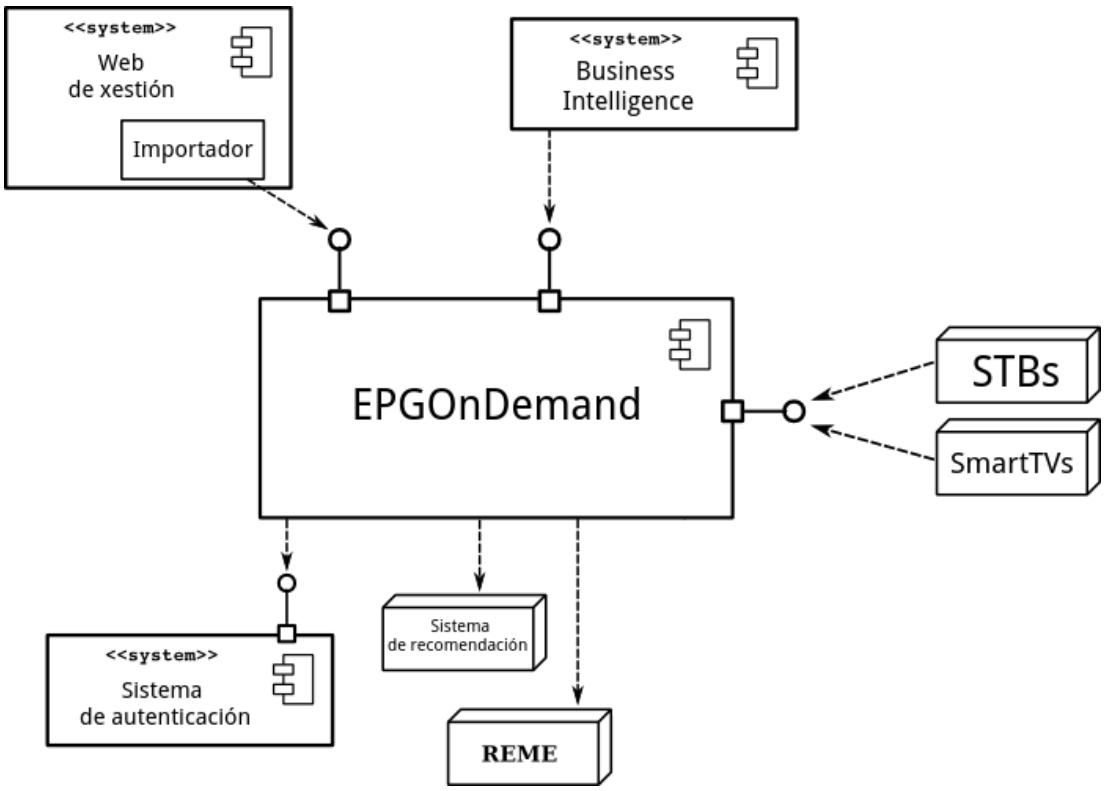

Figura 1.1: Compoñentes de EPGcloud

- **EPGOnDemand**. É o subsistema encargado de proporcionar mediante unha interface de programación REST os contidos almacenados na plataforma (información sobre series, películas, horas de emisión, biografías do reparto dunha produción...) ás aplicacións que fan uso baixo demanda delas. Os datos de EPG proceden dunha base de datos despregada na mesma nube que EPGOnDemand, da que se falará nos seguintes capítulos.
- **Servidor de Medios (REME)**: Repositorio onde son almacenadas as imaxes da plataforma. Oferta unha API para a solicitude de versións específicas das imaxes. Por cuestións empresariais, este servizo non será despregado na nube, polo que será preciso realizar unha proxificación e cacheado do mesmo.

- **Sistema de Business Intelligence (BI)**. A partires dos datos de uso de EPGOnDemand o compoñente de BI realizará tarefas de procesador para facilitar a análise de uso e a facturación aos usuarios e clientes da plataforma.

- **Web de xestión**. Plataforma web interna para a xestión e comprobación da corrección dos contidos que proporciona EPGcloud. Permitirá ademais coñecer o estado da nube de servidores que serven EPGOnDemand (número, carga de CPU) e das bases de datos, así como realizar tarefas de mantemento da base de datos. Ademais, as instancias servindo o software de EPGOnDemand comprobarán se existe unha nova versión do seu software aloxado no nodo de administración, procedendo á súa actualización de ser preciso.

- **Sistema de Autenticación**. Encárgase de realizar a autenticación dos usuarios na plataforma. Unha vez realizado este proceso, o *token* obtido será mantido como válido durante un periodo determinado, evitando deste xeito ter que autenticar o usuario para cada petición. Ademais de para tarefas de facturación, o sistema de autenticación realizará un labor de control de acceso, cunha granularidade que chega até as funcións da API que oferta EPGOnDemand.

- **Sistema de recomendación**. Sistema externo ao que chamará EPGOnDemand para proporcionar recomendacións personalizadas de contido audiovisual en función dos contidos previamente consumidos polo usuario.

- **Importador de datos**. É o sistema encargado de actualizar a base de datos que precisa EPGOnDemand, tomando contido de distintas orixes e formatos. Permitirá importacións incrementais (os datos de EPG son modificados varias veces o día) ou totais (para a carga inicial ou para corrixir algún problema coa actualización incremental).

1.2. Obxectivos do proxecto

O obxectivo principal deste proxecto é, como xa se dixo, proporcionar un API para que os desenvolvedores de aplicacións podan obter metadatos de emisións e

contido baixo demanda de xeito rápido e sinxelo.

Así, partindo do xa sinalado nas seccións anteriores, é posible marcar os seguintes elementos ou metas para o presente proxecto:

- **Desenvolver os elementos de EPGcloud**. En particular, o deseño e implementación de EPGOnDemand e a web de administración do sistema (Figura 1.1). Ademais, desenvolverase o mencionado sistema de almacenamento de *logs* para o futuro servizo de Business Intelligence; situándose éste fóra do alcance do presente proxecto. O Sistema de Autenticación será tamén desenvolvido de xeito externo ao aquí descrito.

- **Acadar unha arquitectura escalable e tolerante a fallos**. As necesidades do proxecto requiren obter un sistema que sexa capaz de adaptarse a mencionada variabilidade da carga, desenvolvendo un modelo flexible e con capacidade asumir caídas de nodos do clúster.

- **Deseñar procedementos de mantemento automático**. Aproveitando as capacidades ofertadas polos sistemas *cloud*, búscase desenvolver procesos de automatización de tarefas de mantemento e comprobación do estado do sistema, sen afectar ao funcionamento de EPGcloud. Os cambios no tamaño do clúster de EPGOnDemand deberán ser automáticos.

- **Desenvolver o sistema de importación da base de datos**. Os datos de EPG serán obtidos da base de datos existente no operador cliente, sendo necesaria unha adaptación ao modelo establecido para EPGcloud. A devandita transformación non será trivial, xa que non será empregado nin o mesmo esquema nin o mesmo software de xestión da base de datos.

- **Realizar probas de carga**. Para garantir que se acadaron os requisitos de rendemento establecidos para a plataforma e coa fin de validar as decisións de deseño que foron tomadas, será preciso realizar probas que leven ao límite o sistema desenvolvido.

1.3. Introdución a Amazon Web Services

Amazon Web Services (AWS) é a plataforma de servizos de computación na nube elixida para despregar EPGcloud. A pesares de que foi expandindo a súa

oferta de módulos, o núcleo inicial de AWS segue o modelo de *Infrastructure as a Service* (IaaS), permitindo ao usuario delegar a xestión do hardware no provedor de servizos. Mediante unha serie de ferramentas web e vía API (REST e SOAP [13]) é posible xestionar os servizos contratados en Amazon Web Services. Deste xeito é posible automatizar tarefas coma o arranque e parada de máquinas virtuais (instancias, na nomenclatura de Amazon Web Services) e xestores de bases de datos, ou a carga de copias de seguridade.

Unha vantaxe fundamental do esquema IaaS é que o usuario só incurre en custos naqueles servizos (capacidade de computación, consumo de largo de banda...) que realmente está empregando; a diferencia dun despregamento tradicional, onde hardware e outras necesidades deben ser mercados -e previstos- por adiantado. Ademais, a elasticidade inherente ao modelo IaaS resulta moi difícil de replicar nun esquema tradicional, pois non resulta sinxelo engadir recursos baixo demanda.

A base prevista de usuarios da plataforma excede os 100.000 dispositivos, polo que a escalabilidade de EPGcloud é unha cuestión da maior importancia durante o desenvolvemento. Ademais, o patrón de uso da plataforma é fortemente dependente da franxa horaria, cunhas *horas val* (periodos de baixa actividade) e picos de carga moi definidos. A posibilidade de modificar o tamaño do clúster de máquinas virtuais servindo EPGOnDemand preséntase coma un factor decisivo á hora de decidirse por un despregamento na nube, polo aforro de custos nos momentos con menor carga e a capacidade de adaptación á demanda naqueles periodos de maior uso.

Amazon Web Services ten unha ampla oferta de servizos *cloud*, desde almacenamento de obxectos (Amazon S3) a bases de datos non relacionais (DynamoDB), pasando por copias de seguridade (Glacier) e computación xeral (EC2). En concreto, os módulos de AWS que se empregarán en EPGcloud son os seguintes:

- **Elastic Compute Cloud (EC2)**. Permite aos usuarios alquilar máquinas virtuais nos centros de datos de Amazon. Estas máquinas poden ser lanzadas e paradas de xeito practicamente inmediato, dotando dunha flexibilidade difícil de acadar nun despregamento tradicional con máquinas físicas. EC2 oferta diferentes *tamaños* de máquinas virtuais, en función dos recursos asignados á instancia. Loxicamente, o custo da instancia increméntase cos cantidade de recursos que esta teña reservados. A posibilidade de des-

pregar máquinas virtuais nos distintos centros de datos de Amazon, xunto ao feito de que o hardware subxacente é xestionado de xeito transparente para o desenvolvedor, confiren unha fiabilidade excepcional.

- **Relational Database Service (RDS)**. Libera ao usuario da xestión da máquina subxacente para servir unha base de datos relacional. Proporciona ferramentas web para xestionar bases de datos MySQL, PostgreSQL, Oracle ou SQL Server. Permite establecer as chamadas *read replicas*, instancias que replican o contido dunha instancia mestra. A sincronización dos datos entre a instancia mestra e as súas réplicas de lecturas faise de xeito automático en Amazon RDS. As instancias de só lectura empréganse coma un mecanismo sinxelo para incrementar a escalabilidade do sistema, xa que poden ser arrincadas ou paradas en calquera momento sen que iso supoña unha parada no sistema. Do mesmo xeito que en EC2, as instancias RDS teñen diferentes tamaños en base aos recursos dos que dispoñen.

- **Elastic Load Balancer (ELB)**. É o balanceador de carga de Amazon Web Services. Permite distribuír as peticións entrantes entre as instancias de EC2 que estean asignadas ao balanceador. Asemade, realiza comprobacións da saúde das instancias, sinalando aquelas que non estean operando correctamente (require establecer unha URL sobre a que facer peticións nas instancias EC2). Amazon Web Services realiza de xeito completamente transparente para o desenvolvedor as tarefas de escalado do propio balanceador de carga, polo que non é preciso realizar ninguna intervención nel a medida que o tráfico aumenta ou diminúe. Ademais, o ELB é quen de traballar como terminador SSL, de xeito que cara ao exterior o tráfico HTTP estea cifrado (HTTPS), mentres que o clúster que atende as peticións traballa en texto plano (co aforro de recursos de computación que isto supón). Por requerimento do cliente, todo o tráfico saínte e entrante de EPGcloud será HTTPS.

- **Cloudwatch**. É un servizo de supervisión e toma de datos dos recursos da nube de AWS, e das aplicacións que se executan en AWS. Empregarase en EPGcloud para obter os datos sobre o estado do clúster para a web de administración e para adaptar o número de instancias de EC2.

- **Auto Scaling**. Combinando información obtida desde Elastic Load Balancer e Cloudwatch, permite lanzar ou parar de xeito automático as instancias EC2 asignadas ao balanceador en función da carga actual do sistema. Para un servizo como EPGcloud, cunhas franxas horarias de uso moi definidas, a posibilidade de adaptar o tamaño do despregamento para adaptarse á demanda precisa supón un importante aforro de custos.

EPGcloud permitirá, pois, enriquecer con información relacionada o contido de entretemento do usuario.

Polas condicións nas que se empregará o servizo, Amazon Web Services preséntase coma unha elección axeitada para o despregamento do proxecto. Mediante as ferramentas proporcionadas por AWS é posible acadar o grao de escalabilidade e tolerancia a fallos que se require nunha plataforma destas características, tanto polo número de usuarios coma pola súa variabilidade, ademais do impacto na experiencia de usuario dunha caída do sistema.

Nos seguintes capítulos detallarase o deseño e a implementación seguidos para EPGcloud.

Capítulo 2

Deseño da arquitectura do sistema

Índice xeral

Neste capítulo definirase a estrutura do despregamento de EPGcloud, particularmente como será instalado nos sistemas que oferta Amazon Web Services. A elección de AWS permite liberar o desenvolvemento de certas tarefas e condicionantes que existen nos despregamentos tradicionais, pero para obter partido das posibilidades que Amazon Web Services ten para a computación distribuída é necesario establecer unha estrutura acorde, coa fin de acadar os beneficios que dito modelo oferta:

- **Escalabilidade**. Deséxase que incrementando o número de nodos do sistema, este sexa quen de dar un servizo a un número maior de usuarios

simultáneos de xeito proporcional á potencia computacional engadida.

- **Tolerancia a fallos**. O obxectivo é que a caída dun nodo non supoña a caída do sistema, pois o resto das instancias deben continuar funcionando (aínda que si pode existir un impacto no rendemento).

2.1. Deseño de EPGOnDemand

O software de EPGOnDemand será despregado en máquinas virtuais de Amazon Web Services (o módulo EC2), coa fin de acadar a mencionada escalabilidade e tolerancia a fallos: por unha banda, será posible modificar automaticamente o número de instancias que atenden as peticións entrantes, incrementando o seu número naqueles momentos de alta carga; pola outra, no caso dunha caída nunha instancia, o sistema seguirá a traballar correctamente.

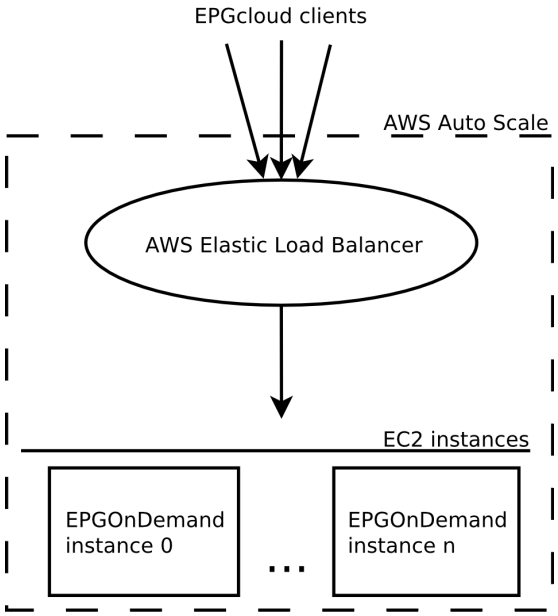

Figura 2.1: Despregamento de EPGOnDemand

Para o correcto funcionamento de EPGOnDemand, tres servizos de AWS deben traballar de xeito coordinado (ver Figura 2.1):

- **EC2**. Proporciona as máquinas virtuais nas que se instalará o software de EPGOnDemand.

- **Elastic Load Balancer**. O balanceador de carga recibirá as peticións dos usuarios da plataforma e as reenviará de xeito equitativo ás instancias de EC2. Realiza tamén controis sobre as máquinas virtuais, para garantir que están funcionando correctamente.

- **Auto Scaling**. Empregando a medición de consumo de CPU das instancias de EC2 (a partir dos datos obtidos desde CloudWatch) asignadas ao balanceador, Auto Scaling decidirá cando é preciso lanzar unha nova instancia, ou parar algunha das existentes. Mediante Cloudwatch estableceranse os límites de tempo e consumo de CPU para realizar tarefas de escalado.

Un despregamento coma este permite escalar dun xeito sinxelo; engadindo novas máquinas será posible atender a máis peticións dun xeito lineal, pois a única interacción que teñen as instancias entre elas é a través da base de datos distribuída que contén información sobre as sesións activas.

2.1.1. Xestión das sesións

O proceso de autenticación contra o Sistema de Autenticación de EPGcloud rematará coa obtención dun *token* que permitirá acceder aos servizos de EPGOnDemand. O devandito *token*, cunha caducidade determinada, definirá a que partes da API de EPGOnDemand ten acceso o usuario. Ademais, servirá para manter información sobre as sesións actuais que poderá ser consultada desde a web de administración.

Para o mantemento das sesións activas empregarase unha base de datos distribuída entre as distintas instancias de EC2 que están a executar o software de EPGOnDemand. Deste xeito, cando unha nova instancia arranque para atender un incremento da demanda, unirase ao clúster de máquinas correndo EPGOnDemand e, de maneira completamente automática, obterá información sobre as sesións activas. Cando o balanceador de carga dirixa unha petición cara a nova instancia, esta xa disporá de información sobre sesións e *tokens*, polo que poderá respostar apropiadamente.

Posto que a base de datos distribuída é un elemento crítico que pode ser un factor limitante na escalabilidade do sistema (as instancias de EPGOnDemand traballan de maneira independente entre elas salvo pola xestión das sesión), deberá ser un software altamente probado en contornos similares.

2.2. Base de datos para EPGOnDemand

Coma se mencionou na introdución, EPGOnDemand proporciona unha API REST para desenvolvedores de aplicacións de diferentes dispositivos, que desexen enriquecer os contidos audiovisuais cos metadatos relacionados co producto (carátulas, argumento, membros do reparto...). Esta información procede na súa meirande parte das bases de datos nos sistemas do cliente que encargou o desenvolvemeto de EPGOnDemand; establecendo tamén o esquema da base de datos.

A devandita estrutura segue na súa meirande parte a das bases de datos existentes, pero sen chegar a ser exactamente igual. Por outra banda, por aforro de custes, soporte da comunidade e rendemento, foi elixido o xestor de base de datos MySQL [15] (incluído en Amazon RDS); cando o empregado nas fontes dos datos é Oracle. Estes dous feitos engadirán complexidade ao sistema de importación, obrigando a traducir as sentenzas de actualización da base de datos dun dialecto de SQL a outro.

2.2.1. Importación de datos de EPG

Os datos a empregar por EPGOnDemand proceden dunha base de datos Oracle cun deseño similar, pero non exactamente igual, ao empregado en EPGcloud. Distinguiranse dous tipos de importación:

- **Incremental**. Será o procedemento habitual de importación de datos de EPGcloud. Un servizo externo subirá os cambios na EPG ao servidor de administración de EPGcloud, sendo aplicados despois sobre a base de datos RDS maestra.
- **Total**. Será o procedemento inicial do sistema, ou en caso de que xurda un problema coa importación incremental. O contido presente e o esquema da base de datos serán borrados, e cargarase unha nova imaxe da base de datos.

A carga das importacións incrementais e totais realizarase desde o nodo de administración. No caso do contido completo da base de datos será un proceso a lanzar de xeito manual desde a web de administración, pois non será un procedemento habitual. As actualizacións incrementais serán automáticas e realizadas varias veces o día.

O sistema de exportación (non descrito no presente proxecto) realizará as tarefas necesarias sobre a base de datos de orixe, coa fin de obter tanto os volcados totais coma os diferenciais para os volcados incrementais. A continuación, transferirá esa información ao nodo de administración.

2.2.2. Despregamento da base de datos en Amazon RDS

Posto que o xestor da base de datos elixido está soportado no *Relational Database Service* (RDS) de AWS, óptase por facer uso deste servizo. Como xa se mencionou na introdución, RDS facilita ao desenvolvedor o despregamento das bases de datos, xa que a administración do sistema subxacente é realizada polo propio servizo.

Por outra banda, dadas as necesidades e a carga prevista sobre a plataforma farase uso das posibilidades de replicación de RDS, particularmente das chamadas réplicas de lectura (*read-replicas*). Este sistema permite definir unha instancia de RDS coma mestra e unha serie de escravos de só lectura, que serán automaticamente sincronizados polo sistema.

A posibilidade de crear até cinco réplicas de só lectura permite escalar o despregamento dun xeito sinxelo, no caso de que ao número de peticións a EPGOnDemand leve ao límite o despregamento presente.

2.3. Proxificación do servidor de medios

Coma xa se mencionou na introdución, e por motivos corporativos, REME (o servizo encargado de proporcionar carátulas, fotografías de persoas do reparto...) non será despregado na nube; estará instalado nun centro de datos tradicional dentro da infraestrutura dun provedor de telecomunicacións.

Ademais, por motivos de seguridade, os administradores do centro de datos no que está despregado REME requiren que só un conxunto establecido de direccións IP poidan acceder desde a nube de EPGcloud. Coma as instancias de EPGOnDemand son susceptibles de cambiar de IP (pois son lanzadas ou paradas en función da carga) é preciso incoporar un ou varios servidores intermedios que manteñan estáticas as IPs (empregando o servizo gratuito Elastic IP de Amazon Web Services), e que sexan estes os que accedan realmente a REME.

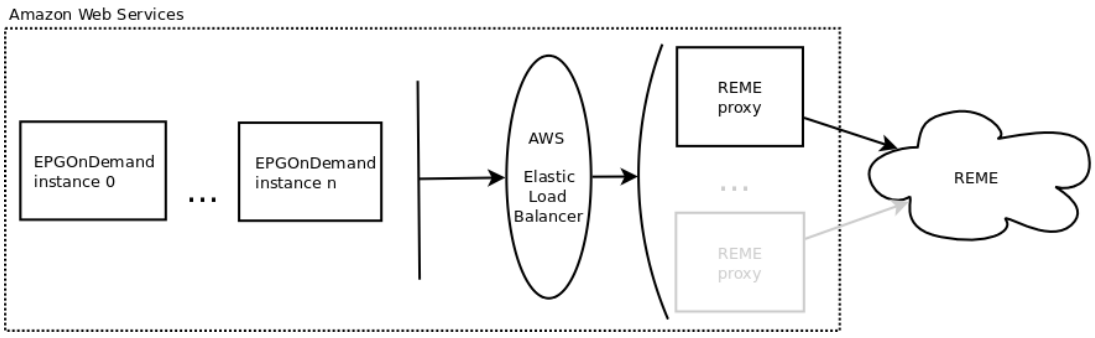

Figura 2.2: Proxificación de REME

Por outra banda, dado que é esperable que exista unha forte semellanza nos contidos consumidos entre os distintos usuarios, a posibilidade de cachear as imaxes de REME en EPGcloud nos mencionados servidores pode conlevar un importante aforro de custes computacionais (nos servidores de REME) e de transferencia cara AWS.

Finalmente, no caso de que fora preciso ter máis dun nodo proxy, instalaríase un balanceador de carga entre os nodos de EPGOnDemand e as instancias de proxificación de REME.

2.4. Servizo de Business Intelligence para a facturación

O sistema de Business Intelligence de EPGcloud precisa de información sobre o número de peticións e tamaño da resposta que realizou un usuario durante un determinado periodo de tempo. Para tal fin deberase almacenar información sobre todas as peticións realizadas sobre EPGOnDemand.

A solución elixida deberá permitir escribir, nun lugar centralizado, información sobre un gran número de peticións por segundo, e desde as múltiples instancias de EC2 onde se está a executar o software de EPGOnDemand. Ademais, deberá facilitar a consolidación ou exportación dos *logs* obtidos.

Optarase por unha solución MySQL aloxada (do mesmo xeito que a base de datos de EPGOnDemand) en Amazon RDS; logo de comprobar o elevado número de insercióǹs por segundo que é capaz de ofertar a instancia RDS máis básica (ver as probas realizadas na sección 4.3).

2.5. Nodo de administración

O nodo de administración non é unha parte crítica de EPGcloud (EPGOnDemand pode seguir a respostar peticións aínda que non estea funcionando), pero si é necesaria para realizar tarefas coma a actualización da base de datos que emprega EPGOnDemand ou o manexo da web de administración. Ademais, os nodos de EPGOnDemand ao seren lanzados comprobarán contra o nodo de administración se está dispoñible unha versión máis recente do software de EPGcloud. Os servizos que prestará este nodo de administración son os seguintes:

- **Servidor de FTP**. Será empregado polo sistema de exportación de datos do cliente da plataforma para transferir os volcados totais e parciais da base de datos de orixe. O nodo de administración deberá ter asignada unha IP estática.
- **Servidor SSH**. Utilizarase para tarefas de administración e para que os nodos de EPGOnDemand comproben se existe unha nova versión do seu software.
- **Web de administración**. Permitirá realizar tarefas de xestión e comprobación do estado do sistema.
  - Xestión da base de datos de EPGOnDemand.
  - Información sobre o estado do clúster e da base de datos.
  - Datos sobre as sesións abertas na plataforma.
  - Consultas á API de EPGOnDemand para verificar a validez dos datos que se están a enviar aos usuarios.

2.6. Visión global

A continuación móstrase o modelo global do despregamento de EPGcloud en Amazon Web Services.

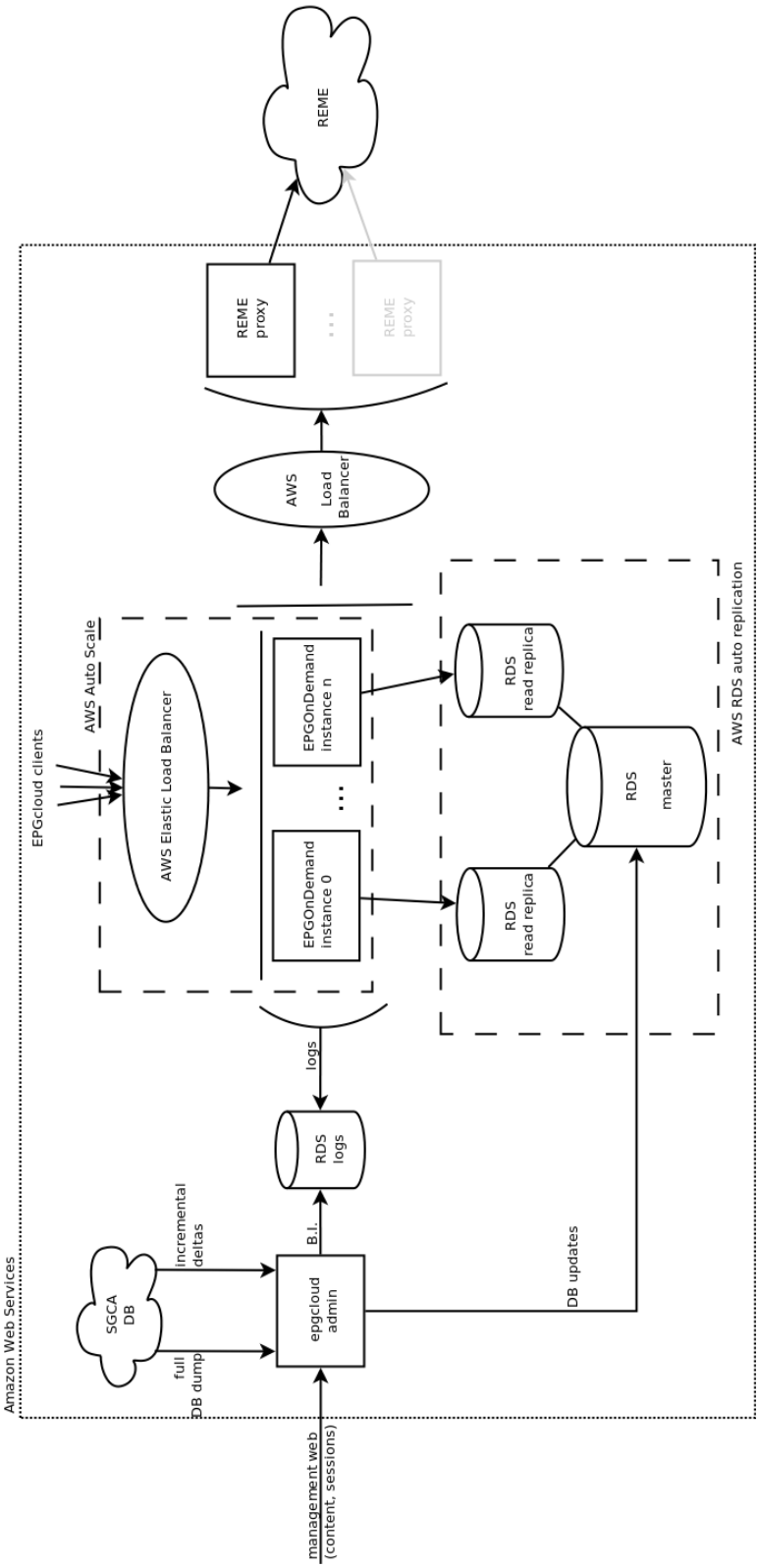

Figura 2.3: Visión global do despregamento de EPGcloud

Capítulo 3

Implementación de EPGcloud

Índice xeral

Neste capítulo realizarase unha explicación do proceso de implementación de EPGcloud e da estrutura, establecida na sección anterior, do seu despregamento en Amazon Web Services. Mostraranse tamén as fases dalgúns dos procedementos automáticos desenvolvidos.

3.1. Implementación de EPGOnDemand

A implementación de EPGOnDemand pode dividirse a súa vez noutras dúas seccións. En primeiro lugar temos o desenvolvemento de EPGOnDemand en si mesmo, é dicir, o software que atende as peticións REST entrantes e realiza as consultas á base de datos de EPGcloud para construír as respostas. Por outra banda, están os elementos auxiliares do sistema, coma a proxificación de REME ou o nodo de administración.

3.1.1. Desenvolvemento do software de EPGOnDemand

A linguaxe elixida para desenvolver o software de EPGOnDemand é Erlang [14]. Erlang é unha linguaxe funcional, baseada no modelo de actores, e que conta de xeito nativo con notables facilidades para a concorrencia e a distribución de tarefas de xeito tolerante a fallos, incluíndo a transparencia para o programador á hora de tratar con procesos locais ou remotos.

Cada petición entrante será atendida por un *GenServer*, un proceso da máquina virtual de Erlang (non confundir coa instancia de EC2 na que se executa o software de EPGOnDemand), traballando de xeito asíncrono coa libraría de acceso á base de datos. Así, aínda que existise un problema coa petición (ben por erro do sistema da base de datos ou por algún problema en EPGcloud), sempre se obterá un resultado en forma de resposta HTTP. Outro proceso da máquina virtual encargarase de almacenar, tamén de xeito asíncrono, a información sobre a petición na base de datos de *logs* do sistema, coa fin de ser consumida polo módulo de Bussiness Intelligence.

O servidor web empregado para servir as peticións, e desde o que se lanzan os mencionados *GenServers*, é *Yaws* [19]; unha solución Erlang moi empregada. As peticións aténdense a través do *framework* proporcionado por *Yaws*; accedendo a base de datos MySQL de RDS a través da libraría *erlmysql* [3].

3.1.2. Pasos no arranque dunha nova instancia

Unha vez o sistema de autoescalado detecta que se están a superar as marxes de consumo de CPU no global de instancias asignadas ao balanceador de carga (establecido, mediante Cloudwatch, no 80 % durante máis de 10 minutos), procederase a arrancar unha nova instancia coa imaxe do software de EPGOnDemand.

1. Inicio da instancia.

2. Comprobación de se o software da instancia é a última versión dispoñible da plataforma, accedendo ao repositorio que se atopa no nodo de administración. Se é preciso, actualizar o software.

3. Arrancar o software de EPGOnDemand.

4. Empregando a API de Amazon Web Services, comprobar se existen máis instancias no balanceador de carga. De ser así, obter as súas IPs internas e rexistrarse na base de datos distribuída *mnesia* [11] (utilizada para manter o estado das sesións). Se é a primeira instancia en acceder ao balanceador de carga, inicializar a base de datos distribuída.

5. Tras un minuto de comprobación por parte do balanceador de carga de que a nova instancia está en correcto funcionamento, incorporala á atención de peticións.

Cando o consumo volva baixar do 30 % de CPU para o global das instancias durante 10 minutos, apagarase unha das máquinas virtuais. Deste xeito, o tamaño do clúster adáptase a carga do sistema, co aforro de custes que isto supón.

Base de datos de sesións

Coma xa se falou nos capítulos anteriores, coa fin de evitar ter que realizar a autenticación do usuario para cada petición a EPGOnDemand, o Sistema de Autenticación devolverá un *token* cun periodo de validez determinado. Asociado a este *token* atópanse o conxunto de métodos da API de EPGOnDemand que ten autorizados o usuario.

Optouse, coma tamén se mencionou, por unha base de datos distribuída. O software da base de datos encargarase de que todos os nodos de EPGOnDemand vexan na súa base de datos local a mesma información, encargándose da sincronización e consistencia.

A base de datos elixida para dar soporte a esta funcionalidade é *mnesia*, unha opción extensamente probada en entornos Erlang, e que mesmo se atopa implementada nesa linguaxe, formando parte das súas librarías estándar (OTP [12]). En *mnesia* a estrutura baséase en tuplas (chave, valor), aínda que o *valor* pode ser heteroxéneo (pode ser mesmamente unha n-tupla). No caso que nos atopa o campo *valor* estará formado pola seguinte información:

- Estado da sesión.
- Data de creación e caducidade.
- Historial de chamadas á API, cos seus correspondentes argumentos.

- Usuario asociado.
- Datos sobre a aplicación cliente (IP, axente de usuario, refer HTTP).
- Información sobre as resposta (tamaño, tempo de procesado, código HTTP).

Cando un novo nodo de EPGOnDemand é lanzado para responder a un aumento da demanda do sistema, esta nova instancia debe entrar a formar parte do clúster de *mnesia*; obtendo deste xeito os datos das sesións actuais. Coma as direccións IP do resto de nodos de EPGOnDemand non son coñecidas (van cambiando a medida que arrancan ou paran as máquinas virtuais ao longo do día), é preciso traballar coa API de Amazon Web Services para descubrir que instancias están activas nese intre tras o balanceador de carga.

Unha vez descuberto o conxunto de direccións IP das máquinas do clúster de EPGOnDemand, procederase a realizar unha operación `net_adm:ping/0` de Erlang a unha delas. Mediante os mecanismos internos da linguaxe, a nova instancia será automáticamente coñecida polo resto delas, que xa estaban en contacto entre elas, e procederase á sincronización da base de datos distribuída.

3.1.3. Desenvolvemento dos elementos de control de EPGOnDemand

Os elementos de control *cloud* de EPGOnDemand e EPGcloud están desenvolvidos empregando Python [16] e a libraría oficial desta linguaxe para a xestión de Amazon Web Services, Boto [9]. Derivado do uso desta utilidade en EPGOnDemand foi posible contribuír con parches aceptados na libraría (ver Sección 5.2).

O elemento de control máis destacable é o proceso mediante o que, no arranque dunha instancia, a API de Amazon Web Services é consultada para descubrir as direccións IP das máquinas virtuais que se atopan tralo balanceador de carga que serve EPGOnDemand, coa fin de unir a nova instancia ao clúster de EPGOnDemand. Coma se mencionou no apartado anterior, unha vez obtidas as direccións das mencionadas máquinas, avisarase ao proceso Erlang encargado para realizar a unión efectiva á base de datos *mnesia*.

De xeito análogo, desenvolveuse un sistema mixto Python-Erlang para aquelas situacións onde sexa preciso realizar un cambio na base de datos que está a

empregar EPGOnDemand. O sistema en Python comunicarase ca API de Amazon Web Services para, chegado o momento, lanzar unha aplicación Erlang que se unirá ao clúster de nodos de EPGOnDemand (nun mecanismo similar ao empregado en *mnesia*) para facer a correspondente chamada remota a unha función de cada un dos nodos, de xeito que se recargue a conexión coa base de datos. O caso típico será a importación dunha nova imaxe completa da base de datos.

Relacionado con este punto atópase a posibilidade de realizar a carga completa dunha nova imaxe da base de datos sen deixar de servir os datos anteriores. A idea básica, expandida nun apartado posterior, é replicar o esquema actual do despregamento en Amazon RDS e cargar nesa nova estrutura os datos desexados.

Ademais, foi preciso engadir un servicio REST para a comprobación do estado da instancia, que sexa accesible desde o balanceador de carga, de xeito que este poida chamalo para verificar que o nodo está funcionando correctamente. Atópase na ruta *http://IP:PORT/EPGOnDemand/health* e antes de volver un código HTTP 200, comproba que é posible realizar unha conexión satisfactoria coa base de datos empregada por EPGOnDemand.

3.2. Proxificación do servidor de medios

Para o acceso a través de proxy do servidor de medios (Sección 2.3) óptase por empregar Varnish [10] coma servidor de proxy e caché. Varnish é un proxy HTTP que tamén realiza funcións de caché. Está altamente probado, e é software libre (con licencia BSD). A súa configuración básica é sinxela: chega con establecer IP e porto de destino (a pública de REME neste caso), e o porto onde escoitará as peticións do lado de EPGcloud.

O proceso seguido para a obtención dunha imaxe será o seguinte:

1. A aplicación cliente realiza a petición dunha imaxe a través da API de EPGcloud.

2. O balanceador de carga traslada a petición a unha das instancias de EPGOnDemand.

3. A instancia de EPGOnDemand comunícase coa máquina virtual que executa Varnish e pide a imaxe empregando a mesma API REST que REME.

4. Varnish comproba se ten a imaxe solicitada na súa caché. Se non a ten, pídella a REME.

5. Unha vez obtida, transfírese á instancia de EPGOnDemand, e de aí cara a aplicación cliente.

Ademais, será preciso asignar IPs estáticas (as mencionadas Elastic IP) ás instancias onde se realice a instalación de Varnish e establecer esas direccións como provedores de REME no ficheiro de configuración de EPGOnDemand. Así, por unha banda, o equipo de seguridade da operadora que xestiona REME poderá permitir o acceso das mencionadas direccións IP a través do seu *firewall* e, pola outra, os nodos de EPGOnDemand saberán a través de que máquinas deben acceder ao servidor de medios.

Finalmente, engadirase coma valor de configuración dos nodos de EPGOnDemand a IP ou IPs onde estea instalado Varnish.

3.3. Despregamento do nodo de administración

Aínda que a instancia de administración non é necesaria para o funcionamento do servizo de EPGOnDemand (ver Apartado 2.5), si é precisa para a actualización da base de datos empregada polo sistema. Serve ademais para realizar labores de mantemento e comprobación dos datos que se están a servir xa que, ao proceder de múltiples orixes e procedementos (incluída a introdución manual), é posible que existan erros.

Despregarase nunha instancia de Amazon EC2 cunha dirección IP estática, coa fin de que os servizos encargados de transferir os volcados da base de datos poidan chegar a ela. Para tal fin, elixiuse *vsftp*, un dos servidores FTP máis empregados para sistemas UNIX, e que conta con licencia GPL.

A máquina virtual de administración tamén realizará a chamada á función `net_adm:ping/0` de Erlang (previa obtención das direccións IP dos nodos de EPGOnDemand a través da API de AWS), de xeito que a web de administración poida mostrar información sobre as sesións da base de datos *mnesia*.

A web de administración está escrita tamén en Erlang, empregando as ferramentas proporcionadas por *Yaws*, o mesmo servidor utilizado para servir a API REST de EPGOnDemand.

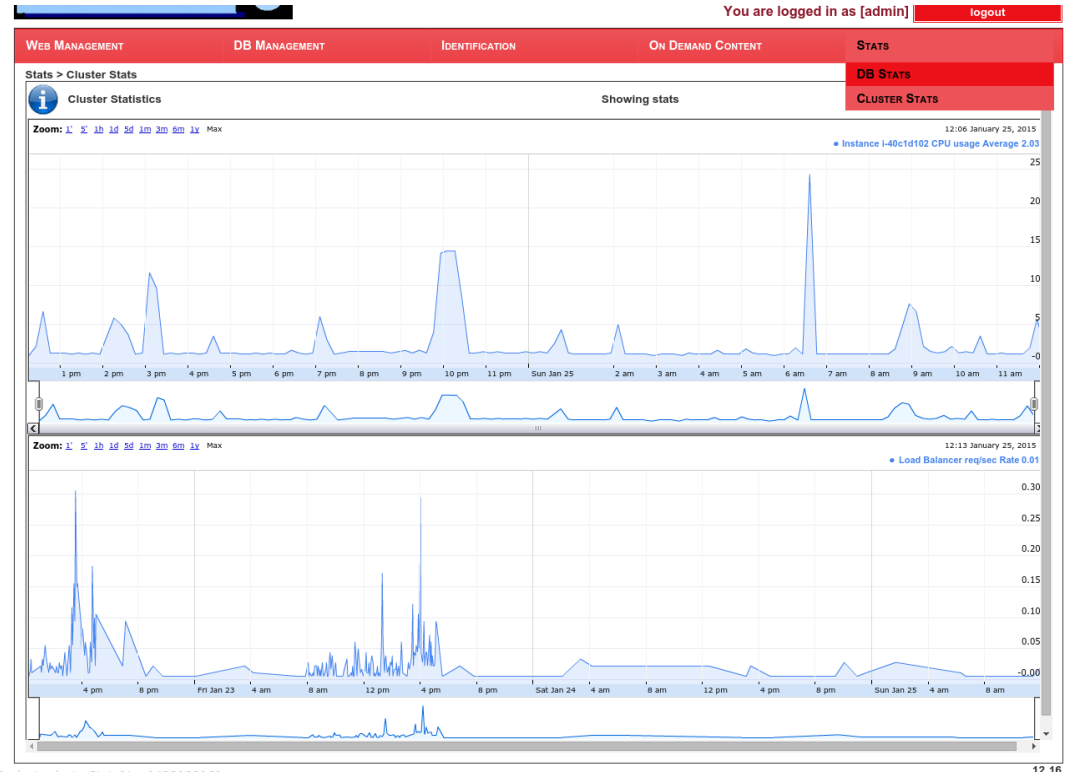

Figura 3.1: Estado do clúster de EPGOnDemand

A información sobre o estado do despregamento en EC2 de EPGOnDemand obtense mediante chamadas á API de AWS Cloudwatch, a través de Boto e de *cloudviz* [5], unha libraría de Python que emprega Google Chart Tools [7] para debuxar os gráficos.

Na Figura 3.1 pódese observar o emprego de *cloudviz* na web de administración para mostrar o estado dun despregamento de EPGcloud; en particular, o uso de CPU da instancia de EPGOnDemand e o número de peticións por segundo que están a ser atendidas polo balanceador de carga.

3.3.1. Carga dunha nova imaxe completa da base datos

Un dos requisitos do cliente da plataforma é que os cambios completos da base de datos non deben afectar a prestación do servizo de EPGcloud (establecido no Apartado 1.2); deberanse seguir servindo os datos anteriores até que a nova imaxe da base de datos estea lista para empezar a funcionar.

Coma xa se mencionou, o proceso que se seguirá é o da creación dunha réplica

do despregamento actual (coa instancia mestra e o mesmo número de réplicas de lectura), cargar o novo *backup* e unha vez listo, cambiar a base de datos que se está a empregar en EPGOnDemand.

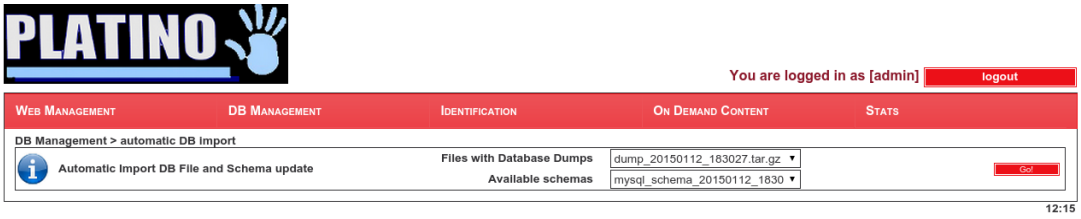

Figura 3.2: Importación dun volcado completo da base de datos

Desde a web de control (Figura 3.2) lanzarase o proceso desenvolvido en Python para proceder coa carga da nova imaxe da base de datos. Os pasos seguidos son:

1. Creación dunha copia de seguridade da base de datos actual.
2. Cambiar o nome de *host* da base de datos mestra actual (o sistema seguirá funcionando pois a dirección IP non cambia).
3. Crear unha nova instancia mestra co nome de *host* da base de datos orixinal, cos mesmos parámetros que a orixinal.
4. Carga do esquema da base de datos.
5. Carga do novo contido da base de datos.
6. Creación das novas réplicas de lectura, se existen no despregamento orixinal.
7. Unha vez comprobado que a nova base de datos está funcionando correctamente, notificarase as instancias de EPGOnDemand que refresquen as súas conexións coa base de datos, de xeito que obteñan as direccións IP da nova infraestrutura.
8. Eliminación da base de datos mestra vella e da súas réplicas (se éstas existían na estrutura orixinal).

O tempo requerido para levar a cabo o proceso pode chegar a alcanzar a media hora, dado o tamaño das táboas da base de datos. En ningún momento do proceso será interrumpido o servizo proporcionado por EPGOnDemand.

Capítulo 4

Análise do rendemento e escalabilidade

Índice xeral

Neste capítulo mostraranse as probas de rendemento e escalabilidade realizadas para verificar o correcto comportamento de EPGcloud e validar as decisións de deseño do despregamento tomadas.

4.1. Características das instancias de proba

Todas as probas foron realizadas sobre máquinas virtuais de Amazon Web Services, posto que esta é a plataforma obxectivo do proxecto. As devanditas instancias son independentes do hardware subxacente, até o punto de ter establecido unha unidade propia de medida de potencia de procesado, o ECU (*EC2 Compute Unit*) [1]; definido por Amazon como a capacidade equivalente "a unha CPU Opteron ou Xeon de 2007 traballando a 1.0-1.2 GHz".

Para realizar as probas de rendemento empregáronse as seguintes instancias de AWS de propósito xeral:

- **t1.micro**. 1 ECU, 613 MB de RAM.
- **m1.small**. 1 ECU, 1.7 GB de RAM.
- **m1.medium**. 2 ECU, 3.75 GB de RAM.
- **m1.large**. 4 ECU, 7.5 GB de RAM.

Tanto RDS coma EC2 empregan a mesma denominación e características para as súas instancias.

4.2. Probas de rendemento e de carga

Para as seguintes probas defínense dous escenarios baseados na experiencia previa do operador cliente de EPGcloud. Este conxunto de chamadas á API de EPGcloud busca replicar do xeito máis realista posible a interacción dun usuario desde o seu *set-top-box*. Estes escenarios serán executados múltiples veces de xeito concorrente, para simular o acceso de distintos usuarios a EPGcloud.

- **Escenario 1**. Usuario navegando pola guía de programación.
  1. Autenticación do usuario.
  2. Obtención da lista de canles.
  3. Acceso á información dunha canle (repítese tres veces para emular o *zapping*).
  4. Obtención da información da emisión actual dunha canle.
  5. Recuperación da imaxe asociada á emisión da canle.
  6. Obtención da descrición longa do evento.
  7. Obtención da información da seguinte emisión da canle.
  8. Recuperación da imaxe asociada á seguinte emisión da canle.
  9. Obtención da descrición longa do seguinte evento.
  10. Acceso á información extendida sobre unha persoa asociada ao devandito evento.
  11. Obtención da imaxe asociada á mencionada persoa.

- **Escenario 2**. Usuario empregando un buscador de eventos de programación.

  1. Autenticación do usuario.
  2. Buscar produto por título.
  3. Obtención dos detalles do produto.
  4. Recuperación da imaxe asociada ao produto.
  5. Obtención das futuras emisións dese produto.

4.2.1. Rendemento de EPGcloud

Neste apartado compararase a execución concorrente dos dous escenarios definidos na sección anterior sobre os diferentes tamaños de instancia de Amazon Web Services, estudando a variabilidade dos resultados. Búscase atopar o tempo medio de resposta baixo unhas condicións de carga realista pola base de usuarios que se agarda, sen chegar a saturar a instancia.

Na Figura 4.1 móstranse os resultados ao realizar 100 peticións concorrentes dos dous escenarios por separado, agardando 200 milisegundos entre petición e petición. En todos os casos empregouse unha base de datos RDS de tamaño *m1.medium*.

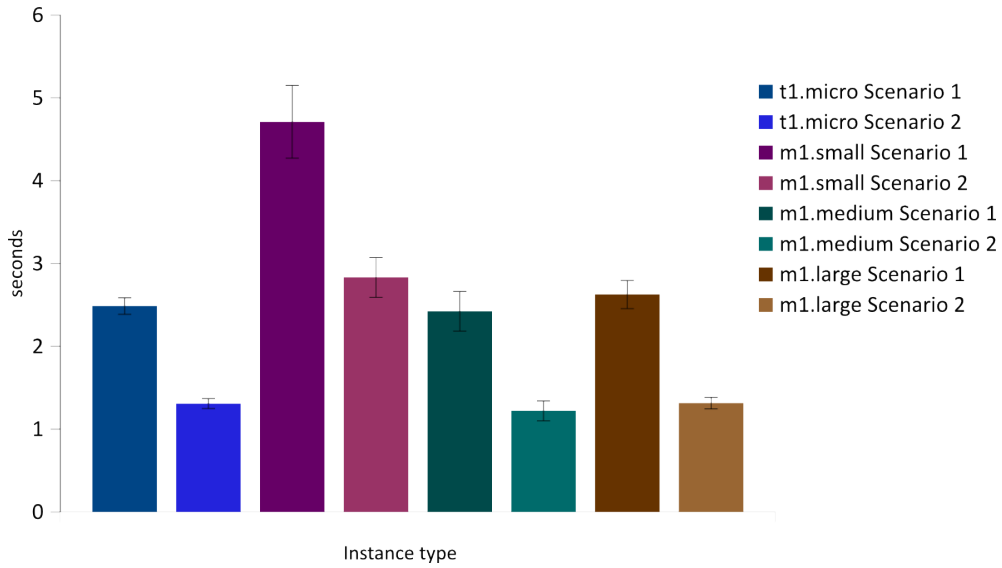

Figura 4.1: Tempo medio de resposta sobre distintos tamaños de instancia

Dos resultados obtidos pódense extraer as seguintes conclusións:

- A instancia *t1.micro* supera aparentemente en rendemento á *m1.small*. Isto é debido a que as instancias *t1.micro* de Amazon Web Services contan cunha capacidade reservada de procesamento limitada, pero que se pode ver incrementada ao obter *bursts* (ráfagas) de CPU sobrante na máquina física [4]. No caso das *m1.small* isto non ocorre; a cambio, segundo Amazon, de obter un rendemento máis consistente.

- As instancias *t1.micro*, *m1.medium* e *m1.large* teñen un rendemento moi semellante (uns 2.5 segundos para o primeiro escenario e 1.25 segundos para o segundo). Isto é debido a que o factor limitante neste caso é a base de datos e a propia conexión con Amazon Web Services a través de Internet.

- En liña co punto anterior, existe unha reducida variabilidade nos tempos de resposta, o que indica que as peticións están sendo atendidas e procesadas sen saturación de ningún tipo.

4.2.2. Resposta á carga de EPGcloud

Nesta sección comprobarase cal é a capacidade máxima de usuarios concorrentes para cada un dos tamaños de máquinas virtuais. Nas Figuras 4.2 e 4.3 móstrase a evolución dos tempos de resposta de cada un dos escenarios, nunha situación onde se busca a saturación ao lanzar un número ilimitado de peticións agardando só 100 milisegundos entre cada unha delas. O momento no que a instancia é saturada está sinalado pola interrupción das liñas.

Pódense observar os seguintes feitos:

- De novo, a instancia *t1.micro* é significativamente máis rápida que a *m1.small*, resistindo máis peticións antes de saturarse no caso do Escenario 1 (Figura 4.2).

- No Escenario 2 (Figura 4.3) a instancia *t1.micro* non chega nin a mostrar indicios de saturación ao recibir peticións cada 100 milisegundos. O mesmo sucede coas instancias *m1.medium* e *m1.large*.

- O Escenario 1 (Figura 4.2) non bloquea as instancias *m1.medium* nin *m1.large*, aínda que na primera si comeza a existir unha certa saturación a partires das 400 peticións.
- A instancia *t1.micro* satura na Figura 4.2 sen chegar aos elevados tempos de resposta que si acada a *m1.small*. Isto é debido a que esgota a súa reducida memoria RAM (630 MB) mentres a súa CPU aínda está a recibir suficientes ráfagas sobrantes da máquina física.
- A instancia *m1.small* ten importantes problemas de rendemento nos dous escenarios.

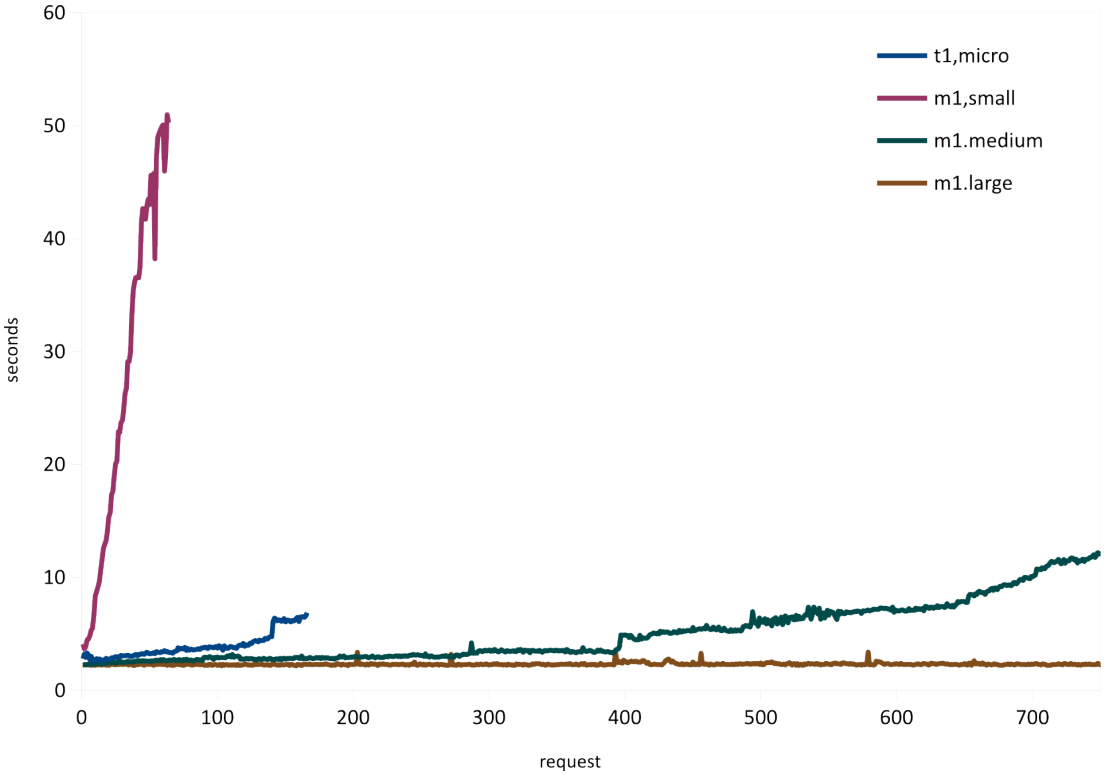

Figura 4.2: Saturación no Escenario 1

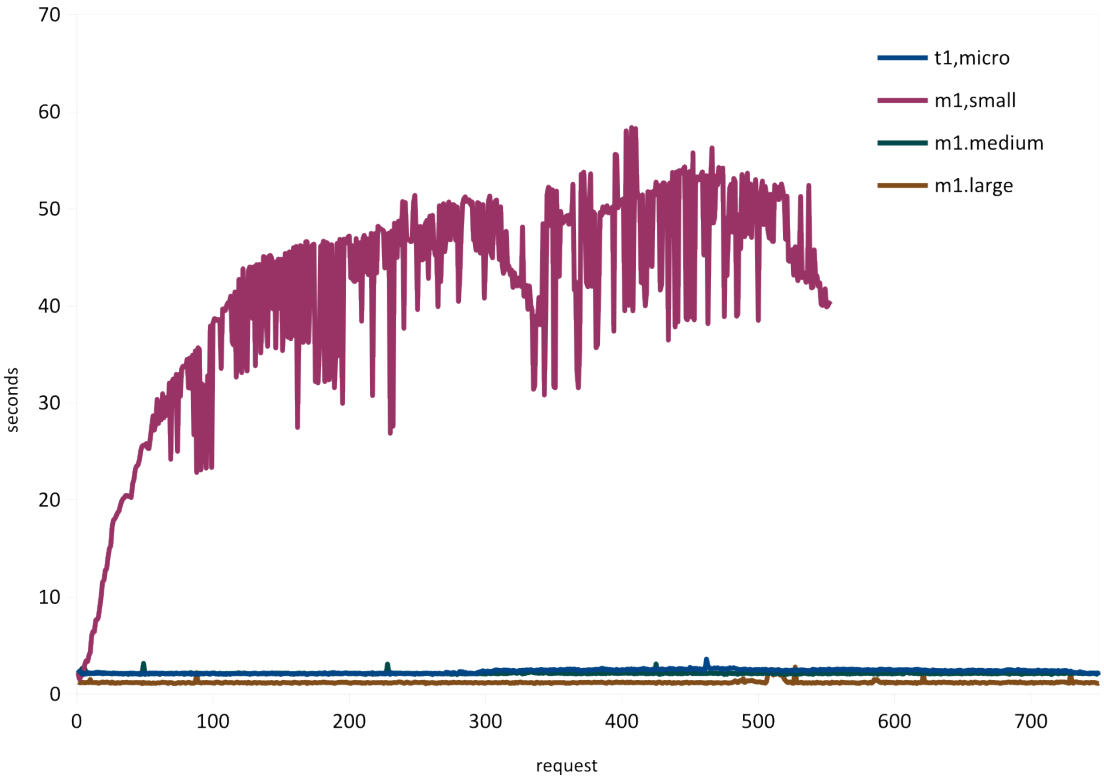

Figura 4.3: Saturación no Escenario 2

4.3. MySQL para o sistema de *logs* de Business Intelligence

Como xa se mencionou na Sección 2.4, coa fin de validar a decisión de escoller MySQL para almacenar os *logs* do sistema de Business Intelligence (ver Figura 1.1), foron realizadas unha serie de probas de inserción de datos en MySQL sobre instancias RDS. A idea é replicar as insercións de liñas de *log* que xeran as instancias de EPGOnDemand logo de cada petición atendida.

Para esta proba empregouse o programa *iiBench* [2], unha aplicación moi utilizada para verificar o rendemento de instalacións de MySQL. Configurouse de xeito que realizase insercións sobre unha base de datos sen máis índices que o empregado polo identificador da táboa, coa fin de replicar o esquema que se empregará na base de datos a utilizar polo sistema de Business Intelligence.

Á vista dos resultados da Figura 4.4 cabe sinalar varios puntos:

- Mesmo a instancia máis modesta de RDS (*t1.micro*), é quen de proporcio-

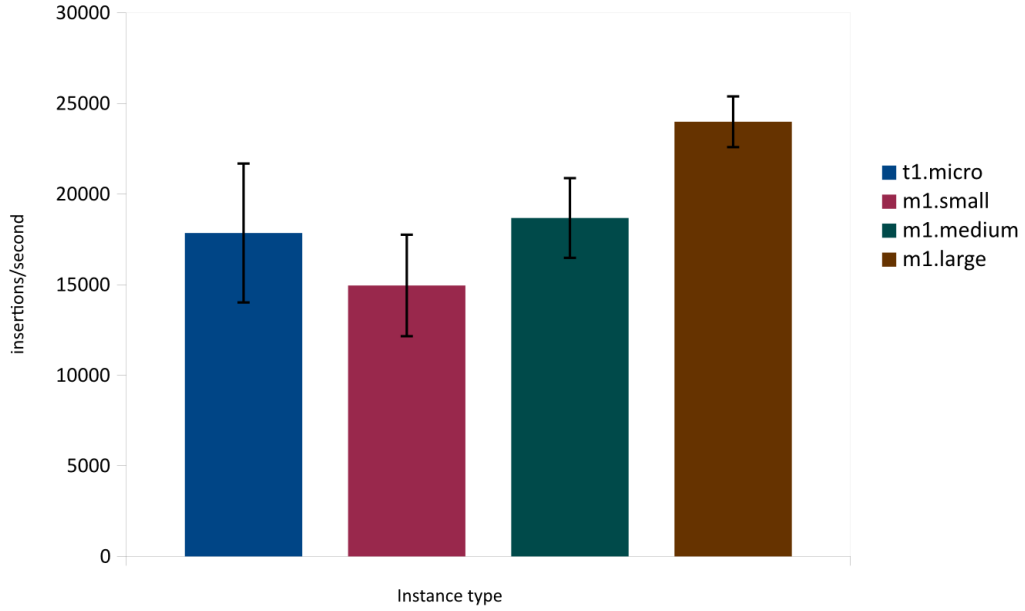

Figura 4.4: Inserciónss por segundo en MySQL con distintas instancias RDS

nar un número de inserciónss por segundo suficiente para non ser o factor limitante no despregamento de EPGcloud.

- A instancia *t1.micro* supera de novo en rendemento á instancia *m1.small* nesta proba sintética.
- Obsérvase tamén unha reducción na variabilidade do rendemento nas instancias *m1.medium* e *m1.large* debida a que Amazon garante para estas instancias un acceso de entrada/saída máis estable.

Deste xeito queda validada a decisión de empregar un sistema centralizado para almacenar os *logs* do sistema, facilitando tanto a implementación da recollida como a súa consolidación para o seu procesado polo sistema externo de Business Intelligence.

Capítulo 5
Conclusións

Os obxectivos principais do desenvolvemento de EPGcloud eran acadar unha plataforma altamente escalable capaz de proporcionar servizo a milleiros de usuarios simultáneos, aproveitando as vantaxes das plataformas IaaS.

Con estas metas en mente, deseñouse a plataforma EPGcloud baseándose nos principios da computación distribuída (escalabilidade e tolerancia a fallos), buscando acadar un modelo capaz de soportar un gran número de peticións; feito que se puxo a proba na Sección 4.2.2.

Establecendo instancias dun tamaño razoable (a partires de *m1.medium* e *m1.large*) é posible respostar sen problema a máis de 200 peticións individuais por segundo por nodo, como se comprobou ao someter ás instancias a probas de carga, cubrindo sobradamente os requerimentos do cliente. Ademais, a capacidade para adaptar o despregamento en tempo real ás previsibles variacións na demanda garante a posibilidade de escalar o sistema.

5.1. Liñas futuras

A principal liña de traballo futura, por requerimentos do cliente, será traballar na compatibilidade de EPGcloud con outras plataformas IaaS, particularmente OpenStack [8] e Eucalyptus [6]. É intención do cliente estudar a posibilidade de despregar EPGcloud no contexto dunha nube privada.

Será preciso, tamén, deseñar e desenvolver o futuro despregamento da plataforma de Business Intelligence, que fará uso dos rexistros que xa son obtidos pola plataforma.

Finalmente, estudarase a posibilidade de empregar Amazon RDS Aurora, un servizo compatible con MySQL que, segundo cifras de Amazon, é capaz de proporcionar un incremento significativo de rendemento.

5.2. Contribucións ao Software Libre

O desenvolvemento de EPGcloud e, en particular, o emprego de Boto (a libraría oficial de manexo de Amazon Web Services desde Python) levou ao descubrimento de *bugs* e carencias na libraría. Os seguintes parches foron aceptados no repositorio principal de Boto.

- *Added support for CopyDBSnapshot.* Até entón, Boto non implementaba a función da API de RDS para copiar imaxes da bases de datos.

  `https://github.com/boto/boto/pull/1872`

- *Added some missing attributes to DBInstance and DBSnapshot.* Engadíronse atributos faltantes á clase que representa unha instancia de RDS, coa fin de poder realizar a operación de carga dunha copia de seguridade da base de datos sen parada do sistema.

  `https://github.com/boto/boto/pull/1880`

- *create_dbinstance ignores value 0 for backup_retention_period.* Correxiuse un erro que causaba que se ignorase o periodo durante o cal se almacena a copia de seguridade automática dunha base de datos.

  `https://github.com/boto/boto/pull/1887`

- *Changed backup_retention_period type to int.* Relacionado coa contribución anterior, modificouse o tipo `backup_retention_period` a `int`, seguindo a API de AWS.

  `https://github.com/boto/boto/pull/1887`

Foi aceptado tamén un parche en *cloudviz* (a libraría empregada na web de administración para mostrar os gráficos de Cloudwatch), para dar soporte a todas as rexións de AWS (`https://github.com/mbabineau/cloudviz/pull/3`).

Apéndice A
Glosario de acrónimos

API *Application Programming Interface.*

AWS *Amazon Web Services.*

EC2 *Elastic Compute Cloud.*

ELB *Elastic Load Balancer.*

EPG *Electronic Program Guide.*

GPL *General Public License.*

IaaS *Infrastructure as a Service.*

IP *Internet Protocol.*

OTP *Open Telecom Platform.*

OTT *Over-The-Top.*

REME *REpositorio de MEdios.*

RDS *Relational Database Service.*

VoD *Video on Demand.*

Bibliografía

[1] What is an ecu? cpu benchmarking in the cloud.
http://goo.gl/rFujM2, 2010.

[2] Indexed insertion benchmark (iibench).
http://www.tokutek.com/resources/technology/iibench/, 2014.

[3] Mysql client for erlang.
http://erlmysql.sourceforge.net/, 2014.

[4] T1 micro instances.
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts_micro_instances.html, 2014.

[5] cloudviz.
https://github.com/mbabineau/cloudviz, 2015.

[6] Eucalytpus website.
http://eucalyptus.com, 2015.

[7] Google chart tools.
https://developers.google.com/chart/, 2015.

[8] Openstack.
http://openstack.org, 2015.

[9] Python boto online documentation.
http://docs.pythonboto.org/en/latest/, 2015.

[10] Varnish website.
https://www.varnish-cache.org/, 2015.

[11] Ericsson AB.

Mnesia.
Ericsson AB, 2 edition, 2014.

[12] F. Cesarini and S. Vinoski.
Designing for Scalability with Erlang/OTP: Implementing Robust, Fault-Tolerant Systems.
O'Reilly, 1 edition, 2008.

[13] G. Duckett.
SOAP Programming: Questions and Answers.
George Duckett, 1 edition, 2015.

[14] J. Armstrong, M. Williams, and R. Virding.
Concurrent Programming in Erlang.
Prentice Hall, 2 edition, 1993.

[15] J. Zawodny and D. Balling.
High Performance MySQL: Optimization, Backups, Replication, Load Balancing and More.
O'Reilly, 1 edition, 2008.

[16] M. Lutz.
Programming Python.
O'Reilly, 4 edition, 2010.

[17] M. Masse.
REST API Design Rulebook.
O'Reilly, 1 edition, 2011.

[18] P. Gurian.
Video on Demand Definitions.
CreateSpace, 1 edition, 2012.

[19] Z. Kessin.
Building Web Applications with Erlang: Working with REST and Web Sockets on Yaws.
O'Reilly, 1 edition, 2012.