ROSES et l'agrégation Web avancée*

Bernd AMANN†, Jordi CREUS TOMAS†, Nicolas TRAVERS‡, Dan VODISLAV⋆

† LIP6
Univ. Pierre et Marie Curie
Paris VI
firstname.lastname@lip6.fr

‡ CEDRIC
Conservatoire National
des Arts et Métiers
firstname.lastname@cnam.fr

⋆ ETIS
Equipes Traitement de
l'Information et Systèmes
firstname.lastname@u-cergy.fr

May 29, 2010

Abstract

La Syndication Web *via* ses deux formats d'échanges RSS et Atom a littéralement explosé sur le Web. La quantité d'information produit sur la toile augmente en continu dépasse les capacités humaines d'analyse de ces informations. Le projet `RoSeS` propose une plateforme de traitement en continu des flux produits sur le Web. Un langage de publication intuitif permet d'intégrer des transformations des flux pour produire un flux virtuel amélioré. Une optimisation de l'ensemble des requêtes persistantes est nécessaires pour permettre le passage à l'échelle du système `RoSeS`. Cet article propose donc une démonstration de l'intégration des requêtes et l'interrogation continue des flux de Syndication Web.

1 Introduction

La plupart des sites web proposent aujourd'hui des services de syndication RSS/ATOM [1] pour diffuser les changements dans leur contenu et surtout l'apparition de nouvelles informations. Ces services complètent le mode de consultation traditionnel par un navigateur web qui déclenche des requête HTTP à la demande de l'utilisateur (mode pull) par la diffusion d'un flux d'items RSS à travers d'un agrégateur vers les utilisateurs abonnés (mode push). Ainsi, grâce à des agrégateurs RSS comme Google reader (`google.com/reader`), Bloglines (bloglines.com) ou Netvibes (`netvibes.com`), les utilisateurs peuvent créer des espaces d'informations personnalisés (mashups RSS) qui diffusent les titres et résumés d'articles publiés à la une dans le journal électronique `lemonde.fr`, informent de l'apparition de nouveaux vidéos sur le site `youtube.com` et permettent de suivre le blog du Dalai Lama sur son mur sur `facebook.com` et les derniers twits d'un ami sur `twitter.com`.

Malgré l'adoption générale de RSS par les fournisseurs de contenus web et malgré une grande variété d'outils et portails pour la consultation de flux RSS, les services d'agrégation de flux RSS proposés se limitent essentiellement à l'affichage et à l'organisation des flux favoris dans des catalogues thématiques. La seule exception est Yahoo Pipes (pipes.yahoo.com) qui propose un éditeur visuel pour créer des nouveaux flux RSS (mashups) par la composition d'opérateurs spécialisés pour le traitement de flux RSS.

Dans le cadre du projet `RoSeS` (ANR-07-MDCO-011), notre objectif est la définition et l'implantation d'un ensemble de services d'acquisition, d'interrogation, de composition et de personnalisation de flux RSS. Ces services sont fondés sur un modèle et un langage de requêtes RSS déclaratif et indépendant d'une technologie particulière utilisée pour leur implantation. Par exemple, la requête suivante agrège deux flux RSS en un seul flux `RoSeS` avec un filtre défini sur les items du flux "http://www.lemonde.fr/rss/tag/culture.xml" :

```
CREATE FEED FestivalCannes AS
RETURN "http://www.lemonde.fr/rss/tag/culture.xml" AS $lemonde |
       "http://www.telerama.fr/tag/rss/festival-de-cannes.xml")
WHERE $lemonde[DESCRIPTION CONTAINS "Cannes"];
```

*The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ROSES (ANR-07-MDCO-011) "Really Open, Simple and Efficient Syndication"

[1] Dans le reste de l'article nous utiliserons le terme RSS pour désigner à la fois toutes les versions du format RSS ainsi que le format ATOM.

Des exemples de requêtes plus complexes permettant de joindre des informations provenant de plusieurs flux sont décrits dans la section 2.1.

Malgré la simplicité de la requête précédente qui utilise uniquement les opérations d'union et de filtrage, la plupart des agrégateurs RSS actuels (à part Yahoo Pipes) ne permettent pas de formuler ce type de requêtes d'agrégation. Une raison de cette limitation est l'effort nécessaire pour l'évaluation de ce type de requêtes dans un environnement réél où des milliers d'utilisateurs veulent créer des agrégations sur des milliers de flux. Dans le cadre du projet `RoSeS` nous étudions différents problèmes liés au passage à l'échelle dans l'agrégation RSS en terme de nombre de flux, nombre de requêtes d'agrégation et nombre de souscriptions.

Dans cette démonstration nous allons présenter `RoSeS`, un agrégateur de flux RSS composé d'un crawler RSS/ATOM couplé à un moteur de requêtes RSS continues et un module de publication/souscription pour composition de flux. Un objectif est d'illustrer l'utilisation du langage `RoSeS` dans le cadre de deux scénarios d'applications (création d'un espace personnalisé et partage d'informations dans un réseau social).

L'agrégateur `RoSeS` présenté peut être considéré comme un système de publication/souscription qui évalue des collections de requêtes continues pour des flux RSS. Il se situe ainsi dans la grande famille de systèmes de gestion de flux de données (Data Stream Management System - DSMS) comme STREAM [?] et des systèmes de publication/souscription par contenu (Content-Based Publish/Subscribe - CBPS) [?]. Ainsi, nous sommes en train d'adapter des techniques d'optimisation multi-requêtes [?, ?, ?] et d'indexation de souscriptions [?, ?, ?, ?] à notre approche d'agrégation de flux RSS.

La section 2 présente le modèle de publication, de souscription ainsi que le graphe d'opération continu. Nous présenterons dans la section 3 l'architecture du système `RoSeS` en détaillant chacun de ses modules. Enfin, la section 4 détaille la démonstration qui sera effectuée à l'aide de deux scénarios illustrant les capacités du système `RoSeS`.

2 Modèle de publication

Les requêtes de publication de flux et la souscription à ces mêmes publications constituent le principal moyen d'interaction entre l'utilisateur et le système `RoSeS`. Le langage de publication permet de définir des flux personnalisés, en utilisant des flux RSS/Atom du Web ou d'autres publications `RoSeS`. Les souscriptions donnent la possibilité de s'abonner au résultat produit par une publication, en spécifiant les modalités de notification.

2.1 Langage

Le langage `RoSeS` permet de définir une nouvelle publication comme un flux virtuel (vue), obtenu par la composition d'autres flux/publications. Cette composition est basée sur quatre types d'opérations sur les flux: *l'union*, *le filtrage*, *le fenêtrage* et *la jointure*.

Un choix important dans le langage de publication est de *ne pas modifier le contenu des items des flux "d'entrée"* par les opérations de composition. En effet, l'union, le filtrage, le fenêtrage et même la jointure (comme on le verra par la suite) préservent l'item du flux et la composition produit ainsi en sortie un sous-ensemble des items d'entrée. La transformation d'items n'est alors possible qu'au niveau des notifications, à la livraison finale des résultats d'une souscription[2]. La séparation entre la transformation et les autres opérations facilite la réécriture et l'optimisation des requêtes de publication.

La syntaxe d'une requête de publication prévoit trois clauses:

- une clause RETURN, obligatoire, qui spécifie *l'union* des flux/publications d'entrée - qui fournissent donc les items de sortie;
- zéro, une ou plusieurs clauses JOIN, qui spécifient les *jointures* des flux de la clause RETURN avec des *fenêtres* sur d'autres flux;
- une clause WHERE, optionnelle, qui spécifie des conditions de *filtrage* sur les flux des clauses RETURN et JOIN.

Prenons l'exemple d'une publication basée sur l'union et le filtrage:

[2]En réalité, `RoSeS` permet de définir des publications matérialisées, qui acceptent des transformations, mais pour des raisons de simplicité, l'opération de transformation n'est pas abordé ici.

```
CREATE FEED MonCinema AS
RETURN ("http://www.allocine.fr/news.rss" | "http://www.lemonde.fr/culture.rss" AS $lemonde) AS $cine
WHERE $lemonde[DESCRIPTION CONTAINS "Cannes"] AND
      $cine[NOT(DESCRIPTION CONTAINS "Richard Gere" OR DESCRIPTION CONTAINS "Julia Roberts")];
```

La publication *MonCinema* réunit le flux AlloCiné et le flux Culture du journal "Le Monde", spécifiés par leurs URLs. La clause RETURN permet de définir des éventuels alias (variables) sur des combinaisons de ces flux, utilisables par la suite dans les autres clauses. Ici on identifie le flux Culture ($lemonde) et l'union des deux flux ($cine). La clause WHERE définit les filtres sur ces flux, avec une syntaxe inspirée de XPath (WHERE $\$var_1[prédicat_1]$ AND ... AND $\$var_n[prédicat_n]$)[3], où chaque prédicat porte sur les items du flux identifié par la variable. Ici, on filtre le flux Culture pour ne garder que les items concernant le festival de Cannes, ensuite le flux union, pour rejeter globalement les items sur des films avec Richard Gere ou Julia Roberts.

Toute nouvelle publication est enregistrée dans *le catalogue* RoSeS et peut être utilisée par la suite dans la définition d'autres publications. A noter la notion de *collection de flux*, qui offre un moyen puissant de créer des publications sur un grand nombre de flux. L'opérateur *collection*("<*requête*>") retourne la collection de flux qui satisfont la requête dans le catalogue - on peut par exemple demander les flux dont la catégorie est "Cinema". Une collection dans la clause RETURN signifie l'union des flux de la collection.

La jointure se fait toujours dans RoSeS entre un flux et une fenêtre. Une fenêtre permet de définir un contexte temporaire qui permet de joindre un flux en continu avec ce contexte. Deux types de fenêtres sur flux sont possibles: basées sur *la durée* (exemple: les items des deux derniers jours) ou sur *le nombre* (exemple: les 30 derniers items). RoSeS utilise un type particulier de jointure, *la jointure d'annotation*. Pour tout item i du flux, on considère l'ensemble $E(i)$ des items de la fenêtre pour lesquels la condition de jointure est satisfaite. Seuls les items i pour lesquels $E(i)$ n'est pas vide font partie du résultat; ces items ne sont pas modifiés, mais seulement *annotés* avec l'ensemble $E(i)$. Cette annotation n'est pas utilisable par la suite pour le filtrage, mais seulement pour la transformation finale à la notification. L'exemple suivant utilise une jointure pour rassembler des critiques de cinéma relatives aux films de la publication *MonCinema*:

```
CREATE FEED CritiquesMonCinema AS
RETURN ("http://www.blog-de-cine1.fr/news.atom" AS $blog1 | "http://www.blog-de-cine2.fr/news.atom") AS $critique
JOIN WITH MonCinema LAST 3 WEEKS
     ON $critique[TITLE SIMILAR WINDOW.TITLE]
WHERE $blog1[AUTHOR != "Julien Criticon"] AND
      $critique[TITLE CONTAINS "critique"];
```

Les critiques proviennent de deux blogs de cinéma, dont on élimine les items produits par Julien Criticon sur le premier blog et on ne garde globalement que les items parlant de critique. On utilise une fenêtre glissante de trois semaines sur *MonCinema*, afin de cibler les critiques concernant les films récents de ce flux.

La souscription à une publication précise le mode de notification (fichier RSS, mail, etc.), la périodicité, ainsi que l'éventuelle transformation appliquée aux items résultat à travers une feuille de style XSL. L'exemple suivant présente une souscription à la publication *CritiquesMonCinema*, avec notification par fichier RSS rafraîchi toutes les 10 minutes et mise en format RSS à travers la feuille de style RoSeS*2RSSFull.xsl*.

```
SUBSCRIBE TO CritiquesMonCinema
APPLY "Roses2RSSFull.xsl"
OUTPUT FILE "critiques.rss" EVERY 10 MINUTES
```

2.2 Graphe de publication

Les pubications réalisées par l'ensemble des utilisateurs RoSeS sont évaluées continuellement par le système. Le modèle d'exécution est basé sur *l'algèbre* RoSeS, composée principalement des opérateurs d'union, filtrage, fenêtrage et jointure.

Chaque publication peut être représentée sous forme de *graphe d'opérateurs*, le plan logique d'exécution étant obtenu par l'assemblage des graphes de toutes les publications impliquées dans les souscriptions. Ce graphe global est utilisé comme support pour *l'optimisation logique multi-requêtes*, basée sur la réécriture et la factorisation des chemins - ces aspects sont en dehors du cadre de cet article. Le graphe logique optimisé est traduit en un plan physique d'exécution, qui assure une *exécution continue* des requêtes, au fur et à mesure de l'arrivée des items.

[3] Il est également possible de spécifier les prédicats de filtrage directement dans les clauses RETURN/JOIN.

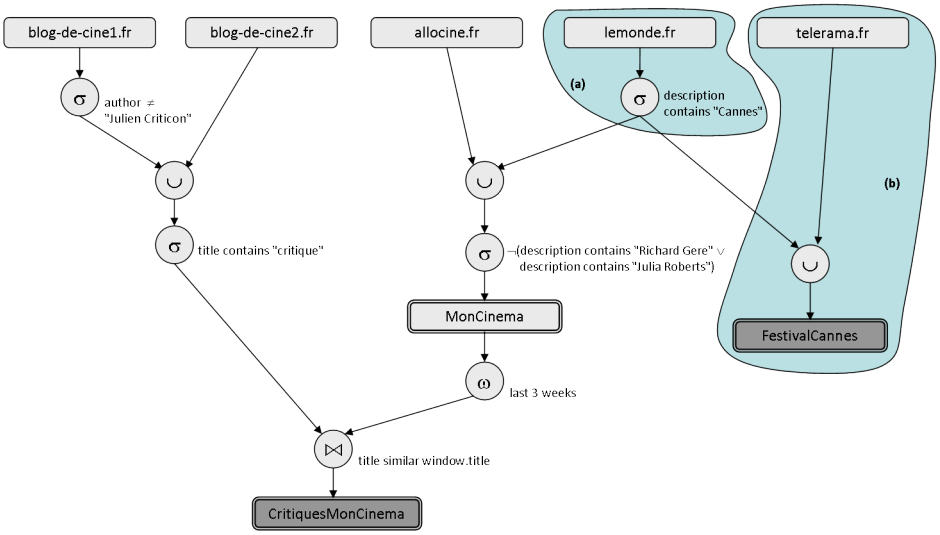

Figure 1: Exemple de plan logique d'exécution

La figure 1 présente le plan logique d'exécution pour deux souscriptions : *CritiquesMonCinema* et *FestivalCannes*. On distingue les graphes d'opérateurs pour les publications *MonCinema*, *CritiquesMonCinema* et *FestivalCannes* (les flux Web d'entrée sont représentés par le nom du site). Une factorisation a été réalisée pour le chemin (a), partagé par *MonCinema* et *FestivalCannes*. A noter que seules les publications concernées par les souscriptions courantes sont présentes dans ce graphe. Dans le cas où la souscription *FestivalCannes* serait annulée, le sous-graphe (b) disparaîtrait du plan d'exécution.

3 Architecture

Une fois les publications sont intégrées dans le graphe d'opérateurs, les items produits y sont alors évalués, pour les souscriptions définies par les utilisateurs. L'architecture du système `RoSeS` est représentée dans la figure 2, nous pouvons constater la décomposition en trois modules (*Acquisition*, *Évaluation* et *Diffusion*) en interaction avec le *Catalog* et le *System Manager*. Nous allons étudier leurs rôles respectifs.

Le **module d'Acquisition** génère un flux d'items à partir de sources dynamiques. Le flux d'items d'une source correspond à la succession des différences de contenu à chaque consultation. En effet, le paradigme de Syndication Web est un mode `Pull`, le module d'acquisition doit alors détecter les changements entre deux consultations de la source, produisant le flux d'items. Ainsi, le but de l'acquisition est de calculer la meilleure *période de rafraichissement*. Elle dépend de la fréquence de mise à jour de la source et du nombre total de flux à rafraichir (partage du temps d'acquisition). Ce rôle est fourni par le composant *Freshness* qui minimise à la fois la perte d'information, et le nombre d'accès aux sources. Un adaptateur (Wrapper) est disponible pour chaque type de source dynamique : RSS/Atom, Service Web, base de données. Le résultat du module d'Acquisition est un flux continu des items `RoSeS` associés à un identifiant de flux correspondant à la source. Ces items `RoSeS` sont le résultat du formatage des items des sources vers un format homogène.

Le **module d'Évaluation** traite en continu les items transmis au *Dispatcher* qui route, au `Graphe Physique` d'opérateur (`GP`), les items en fonction de leur source. Ce graphe est obtenu après transformation du Graphe Logique (section 2.2) auxquels sont associés des *Thread* et des *Files d'Attente* (FA) en entrées et en sorties. Deux problèmes majeurs surviennent sur ce graphe, (1) au vu de sa taille, l'exécution en parallèle de tous les opérateurs n'est pas envisageable, (2) l'ajout et la suppression de requêtes demandent une *Optimisation Dynamique* du *GP*. L'exécution en parallèle des opérateurs est gérée par le `Scheduler` qui défini des priorités parmi les opérateurs suivant deux stratégies : prioriser les opérateurs ayant les *FA* en entrées les plus remplies (gains de place), ou ceux dans les items sont les plus anciens dans les *FA* (validité de la réponse). Le `Runtime Optimizer` s'occupe de l'Optimisation Dynamique du *GP*. Lors de l'insertion d'une publication, il est possible de factoriser un sous-arbre d'opérations identiques

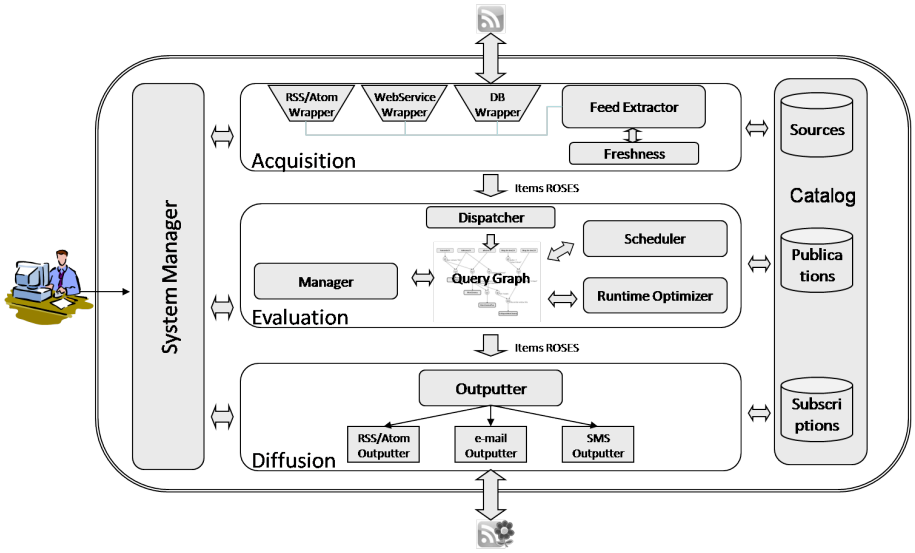

Figure 2: Architecture du système `RoSeS`

au contenu du *GP*. Ce point crucial de l'optimisation permet de réduire la taille du *GP* en créant des *multi-opérateurs* (plusieurs sorties) et ainsi de réduire le nombre de traitements d'un item.

Le **module de Diffusion** transforme les résultats publiés dans le format demandé par les souscriptions des utilisateurs. A chaque format (ou mode) est associé un `Ouputter` spécifique (RSS, SMS, email, appel de service). Le mode de diffusion défini aussi la périodicité à laquelle la notification doit être réalisée. Le rôle du module de diffusion est donc de partager et d'optimiser les ressources de notification entre tous les utilisateurs pour produire des résultats dans les délais requis.

Le **System Manager** permet de contrôler le système `RoSeS`. Le module d'`Acquisition` peut changer d'algorithme de calcul de fraicheur, ajouter ou supprimer des sources, et le nombre de sources à acquérir en parallèle. Le module d'`Évaluation` peut changer la stratégie du `Scheduler`, le nombre de *Threads*, consulter le *GP*, ajouter et supprimer des publications. Le module de `Diffusion` peut consulter, ajouter et supprimer des souscriptions, et modifier la configuration par défaut des notifications.

Le **Catalog** contient les informations indispensables pour les trois modules. L'`Acquisition` stocke les informations concernant les sources, leur type, leur fréquence de mises à jour et d'erreurs. L'`Évaluation` garde les publications sous forme de plans normalisés. Et la `Diffusion` stocke les souscriptions avec leur mode et les paramètres de notification.

4 Démonstration

Nous avons développé un prototype du système `RoSeS` qui permet à l'utilisateur (a) de gérer ses publications `RoSeS` ainsi que ses souscriptions et (b) de consulter les flux d'items générés par les publications et souscriptions. L'interface de gestion permet d'ajouter des nouvelles publications au système et d'en supprimer, mais aussi de les partager avec des autres utilisateurs, elle permet également l'ajout et la suppression des souscriptions aux publications. D'un autre côté, l'interface de consultation permet à l'utilisateur de lire les flux auxquels il a souscri, de consulter les flux produits par ses publications, ainsi que de rechercher des publications définies par d'autres.

Nous proposons une démonstration de notre système avec deux scénarios. Dans le premier cas, nous montrerons l'utilisation de *Collections*, c'est-à-dire un regroupement d'une grande quantité de flux sur une même thématique. Deuxièmement, nous montrerons l'utilisation du système à travers des *filtres contextuels* (opérateurs de jointure), afin de générer des flux qui changent de manière continue de thématique en fonction du contenu des autres flux.

I) *Collections*. Supposons un utilisateur fan d'un joueur de tennis (Rafaël Nadal) et qui veut être au courant de toutes les news sur lui pendant le tournoi de *Roland Garros*, sans toutefois chercher

toutes les sources qui pourraient l'intéresser. Il lui suffit d'utiliser une Collection (*Sport*) définie par un autre utilisateur constituer de l'union d'une centaine de sources sportives. Ainsi, il pourrait définir sa publication, tout simplement, comme suit :

```
CREATE FEED Nadal AS
RETURN Sport
WHERE LANG = "fr" AND ITEM CONTAINS "Nadal" AND ITEM CONTAINS "Roland Garros";
```

II) *Filtrage contextuel.* Supposons maintenant un utilisateur qui utilise plusieurs services de syndication de contenus (Twitter, Flickr, YouTube...), comme ses amis. Il ne regarde pas tous les jours ce que ses amis ont publié, mais il aimerait bien être notifié de leur contenu, mais ayant un rapport avec le contenu de l'utilisateur. Par exemple, s'il écrit sur son Twitter "*je vais au concert de Muse ce week-end !*", il voudrait savoir si ses amis ont publié n'importe où quelque chose sur *Muse* récemment (une vidéo sur YouTube, un post sur leur blog, etc.). Donc, il peut définir une publication qui agrège tous les flux d'information et qui fait une jointure entre les items de ses propres flux et les items des flux de ses amis, comme ça :

```
CREATE FEED MonFluxContextuel AS
RETURN ("http://twitter.com/moi/feed.rss" | "http://www.mon-blog.fr/news.atom" |
        "http://www.flickr.com/moi/photostream.rss" | "http://www.youtube.com/moi/videos.rss") AS $moi
JOIN WITH ("http://twitter.com/ami1/feed.rss" | "http://www.flickr.com/ami1/photostream.rss" |
           "http://twitter.com/ami2/feed.rss" | "http://www.blog-ami2.fr/news.atom" |
           FluxAmi3) LAST 1 WEEKS
ON $moi[WINDOW.ITEM CONTAINS KEYWORDS];
```

Durant la démonstration, nous consulterons l'intégration des requêtes dans le système à l'aide du graphe de requêtes. Nous testerons l'évaluation en continu des requêtes à l'aide du module d'acquisition, mais également grâce à un émulateur de flux produit par l'agrégation de plusieurs milliers de flux durant une longue période (plusieurs mois). Les résultats de l'évaluation pourront être consulter sous forme de flux RSS en fonction des paramètres des souscriptions.