Annonce de la soutenance de thèse de Etienne Brodu

FLUXIONAL COMPILER : SEAMLESS SHIFT FROM DEVELOPMENT PRODUCTIVITY TO PERFORMANCE EFFICIENCY, IN THE CASE OF REAL-TIME WEB APPLICATIONS

Thèse CIFRE effectuée au
Centre d'Innovation en Télécommunication et Intégration (CITI) de l'INSA de Lyon,
en collaboration avec l'entreprise Worldline.
sous la direction de Stéphane Frénot et Frédéric Oblé.

Soutenance publique

21 Juin 2016, à 14h.

Bâtiment Claude Chappe.

Devant le jury composé de

Gaël THOMAS	Professeur des Universités, Telecom SudParis	Rapporteur
Frédéric LOULERGUE	Professeur des Universités, LIFO	Rapporteur
Floréal MORANDAT	Maître de conférences, LaBRI	Examinateur
Frédéric OBLÉ	Docteur, Atos Worldline	Examinateur
Stéphane FRÉNOT	Professeur des Universitéss, INSA Lyon	Directeur

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

citi lab

worldline e-payment services

ABSTRACT

Most of the now popular web services started as small projects created by few individuals, and grew exponentially. Internet supports this growth because it extends the reach of our communications world wide, while reducing their latency. During its development, an application must grow exponentially, otherwise the risk is to be outpaced by the competition.

In the beginning, it is important to verify quickly that the service can respond to the user needs: Fail fast. Languages like Ruby or Java became popular because they propose a productive approach to iterate quickly on user feedbacks. A web application that correctly responds to user needs can become viral. Eventually, the application needs to be efficient to cope with the traffic increase.

But it is difficult for an application to be at once productive and efficient. When the user base becomes too important, it is often required to switch the development approach from productivity to efficiency. No platform conciliates these two objectives, so it implies to rewrite the application into an efficient execution model, such as a pipeline. It is a risk as it is a huge and uncertain amount of work. To avoid this risk, this thesis proposes to maintain the productive representation of an application with the efficient one.

Javascript is a productive language with a significant community. It is the execution engine the most deployed, as it is present in every browser, and on some servers as well with Node.js. It is now considered as the main language of the web, ousting Ruby or Java. Moreover, the Javascript event-loop is similar to a pipeline. Both execution models process a stream of requests by chaining independent functions. Though, the event-loop supports the needs in development productivity with its global memory, while the pipeline representation allows an efficient execution by allowing parallelization.

This thesis studies the possibility for an equivalence to transform an implementation from one representation to the other. With this equivalence, the development team can follow the two approaches concurrently. It can continuously iterate the development to take advantage of their conflicting objectives.

This thesis presents a compiler that allows to identify the pipeline from a Javascript application, and isolate its stages into fluxions. A fluxion is named after the contraction between function and flux. It executes a function for each datum on a stream. Fluxions are independent, and can be moved from one machine to the other, so as to cope with the increasing traffic. The development team can begin with the productivity of the event-loop representation. And with the transformation, it can progressively iterate to reach the efficiency of the pipeline representation.¶

RÉSUMÉ

La plupart des grands services web commencèrent comme de simples projets, et grossirent exponentiellement. Internet supporte cette croissance en étendant les communications et réduisant leur latence. Pendant son développement, une application doit croître exponentiellement, sans quoi elle risque de se faire dépasser par la compétition.

Dès le début, il est important de s'assurer de répondre aux besoins du marché : Fail fast. Des langages comme Ruby ou Java sont devenus populaires en proposant la productivité nécessaire pour itérer rapidement sur les retours utilisateurs. Une application web qui répond correctement aux besoins des utilisateurs peut être adoptée de manière virale. Mais à terme, une application doit être efficace pour traiter cette augmentation de trafic.

Il est difficile pour une application d'être à la fois productive et efficace. Quand l'audience devient trop importante, il est souvent nécessaire de remplacer l'approche productive pour un modèle plus efficace. Aucune plateforme de développement ne permet de concilier ces deux objectifs, il est donc nécessaire de réécrire l'application vers un modèle plus efficace, tel qu'un pipeline. Ce changement représente un risque. Il implique une quantité de travail conséquente et incertaine. Pour éviter ce risque, cette thèse propose de maintenir conjointement les représentations productives et efficaces d'une même application.

Javascript est un langage productif avec une communauté importante. C'est l'environnement d'exécution le plus largement déployé puisqu'il est omniprésent dans les navigateurs, et également sur certains serveurs avec Node.js. Il est maintenant considéré comme le langage principal du web, détrônant Ruby ou Java. De plus, sa boucle évènementielle est similaire à un pipeline. Ces deux modèles d'exécution traitent un flux de requêtes en chaînant des fonctions les unes après les autres. Cependant, la boucle évènementielle permet une approche productive grâce à sa mémoire globale, tandis que le pipeline permet une exécution efficace du fait de sa parallélisation.

Cette thèse étudie la possibilité pour une équivalence de transformer une implémentation d'une représentation vers l'autre. Avec cette équivalence, l'équipe de développement peut suivre les deux approches simultanément. Elle peut itérer continuellement pour prendre en compte les avantages des deux approches.

Cette thèse présente un compilateur qui permet d'identifier un pipeline dans une application Javascript, et d'isoler chaque étape dans une fluxion. Une fluxion est nommée par contraction entre fonction et flux. Elle exécute une fonction pour chaque datum sur le flux. Les fluxions sont indépendantes, et peuvent être déplacées d'une machine à l'autre pour amortir l'augmentation du trafic. L'équipe de développement peut commencer à développer avec la productivité de la boucle évènementielle. Et avec la transformation, elle peut itérer pour progressivement atteindre l'efficacité du pipeline.