The REBOL Documentation Project

-- FR - Documentation REBOL - Divers --

Divers

Le binding à la volée (Just-in-time (JIT))

Philippe Le Goff

Première publication : 20 mai 2006, et mis en ligne le samedi 20 mai 2006

Résumé :

La traduction de l'article 10 du Blog du Rebol 3.0, qui porte sur un nouveau type de binding.

Carl Sassenrath, CTO REBOL Technologies 11-Apr-2006 16:39 GMT Article #0010

REBOL 3.0 va introduire un nouveau type de binding : le binding à la volée (Just-in-Time). C'est un article assez long, mais je pensais que vous seriez intéressés par des détails, et je sais que vous ne serez pas timides en me faisant vos commentaires

Actuellement, REBOL supporte deux sortes de binding :

- Definitional Context Binding (DCB) C'est le binding qui se produit lorsque vous définissez un bloc de données comme le corps d'un datatype particulier ou un contexte. C'est le binding usuel tel qu'il est implémenté par la fonction **bind**. Par exemple, les variables de la fonction sont liées au corps de la fonction lorsque **make function !** est mis en oeuvre (ou les mezzanines comme **func**, **function**, etc.)

- Late Path Binding (LPB) C'est le binding qui se produit lorsque vous utilisez un path contenant des champs symboliques. Ce binding très tardif se produit durant l'évalution du path lui-même. Cela arrive principalement quand vous accédez aux champs d'un objet, et il est effectué tardivement pour que les champs soient eux-mêmes des variables.

Ces méthodes de binding fonctionnent bien pour le code REBOL classique ; cependant, pour traiter des dialectes en REBOL, ils n'apportent aucun bénéfice car les mots du dialecte ne subissent jamais le processus de définition. Les dialectes sont juste des blocs, pas des datatypes spécialisés tels que les fonctions ou les objets.

Durant l'implémentation du projet Rebcode (qui a créé un modèle unique de machine virtuelle REBOL très performante), il est devenu évident qu'une méthode très rapide de consultation était nécessaire pour les op-codes de la machine virtuelle (VM). Lorsque vous indiquez un "add" ou un "sub", vous ne souhaitez pas que l'interpréteur dépense du temps en essayant d'imaginer ce que ces op-codes signifient.

Deux solutions deviennent évidentes :

utiliser le DCB ou créer un nouveau type de binding "tardif", qu'en attente d'une définition plus convenable j'ai appelé "just-in-time binding (JIT)". Les deux méthodes seront à prendre en compte ; cependant, parce que le Rebcode a été incorporé dans un wrapper de fonction, il a eu l'opportunité d'utiliser DCB. En outre, on ne permet pas la modification des blocs de Rebcode après leur définition. Ainsi le DCB était parfait pour cela et a amélioré la vitesse de Rebcode par trois ou quatre fois.

Bien, qu'en est-il à propos des dialectes REBOL ? Les performances des dialectes avec draw et avec parse peuvent-elles être améliorées ?

Oui, c'est possible. Prenez par exemple le dialecte draw. Les mots-clés (commandes) du dialecte sont évalués rapidement, mais pour des grands blocs (comme de grandes images SVG), les performances pourraient être meilleures. Une quantité significative de temps est nécessaire pour résoudre les mots-clés chaque fois que le bloc draw est évalué.

Rebcode a été implémenté en utilisant la technique dans laquelle un contexte pour le dialecte définit les mots du dialecte, et le DCB a été utilisé pour binder le bloc du dialecte à ce contexte. Il n'y a rien

d'extraordinaire jusque là.

Cependant, cette méthode avec le DCB ne marche pas bien avec des dialectes comme draw, qui peuvent changer dynamiquement. Par exemple, durant une animation, le programmeur peut insérer ou supprimer des commandes dans ou depuis le bloc draw.

De tels changements diminueraient la raison d'être du travail effectué par le DCB, sachant qu'une alternative serait de forcer le programmeur à maintenir le binding manuellement pendant chaque insertion, mais que nous préférerions éviter cela.

C'est là qu'entre en scène le "JIT binding", le binding à la volée. Avec le JIT, les mots d'un bloc sont liés au contexte qu'ils ont trouvé durant leur évaluation. Contrairement au LPB, les mots d'un bloc sont modifiés pour conserver leurs nouveaux bindings. Ils sont "collants". A l'évaluation suivante du bloc, un contrôle rapide du contexte des mots est exigé, et aucune autre action n'est nécessaire. Si un nouveau mot a été inséré, le contrôle échoue, et le binding JIT se produit.

Le résultat est que des dialectes avec draw et parse sont évalués plus rapidement, mais sans la nécessité de pre-binder les mots au bloc du dialecte. Et, si le bloc reste globalement peu modifié, la vitesse d'évaluation d'un dialecte est très proche du pur code REBOL (nécessitant seulement un contrôle de contexte pour chaque mot).

Y-a-t'il une contre-partie à cette méthode ? Oui. La bonne conception repose sur la gestion des différences. Cela prend une petite fraction de temps de définir la méthode JIT avant l'évaluation du dialecte. Pour des blocs de petite taille qui contiennent moins de trois ou quatre mots-clés, le coût de la prise en compte de la JIT peut ralentir le traitement de ces blocs. C'est le genre de chose que nous devrons tester et évaluer. Il peut aussi y avoir quelques pénalisations légères pour les exceptions (erreurs, break, retours, thrown) qui se produisent durant l'évaluation des dialectes en mode JIT ; cependant, ces évènements devraient être rares dans la plupart des codes de dialectes.

En conclusion : le binding à la volée (just-in-time) est juste une sorte de DCB effectué "à la volée". Il est possible que ce type de binding puisse conduire à des gains en performance pour les dialectes, qui utilisent **draw**, **parse**, etc. Il est aussi possible que nous utilisions le JIT au lieu du LPB pour améliorer les performances sur les accès des champs des objets. Cela doit être étudié.

Note : ce concept est encore au tout début de la phase de conception, et nous en saurons plus bientôt. Si vous avez des commentaires sur cette idée, je sais que vous serez heureux de les poster sur le blog. Merci.

(Traduction : Philippe Le Goff)