

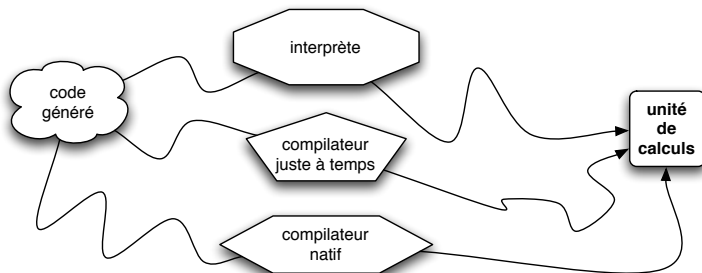
Techniques de génération dynamique de code

Philippe Wang

9 novembre 2006

Architecture logicielles pour l'autoadaptabilité dynamique - UPMC - 2006/2007

La génération dynamique de code consiste à générer du code machine¹ lors de l'exécution d'un programme.



¹ autrement dit, des instructions pour le processeur...

*Réflexions, introspection, intercession, réification,
portabilité, efficacité, ...*

- réification, auto-examination, modification structurelle, modification comportementale
- portabilité : compilation vers du code-octet
 - perte de performance
 - regain de performances : compilation juste-à-temps
- spécialisation de code à l'exécution dans le cadre du polymorphisme

- Compilation juste-à-temps
 - Smalltalk, Java, .Net, OCamlJIT, (\sim C), ...
- Chargements dynamiques de code-octet
 - .Net (C#) : `Assembly.LoadFrom`
 - Objective Caml (ZINC) : Module **Dynlink**
 - ...
- Chargements dynamiques de code source
 - Langages de scripts
(javascript, perl, python, bash, php, ...)
 - Java : `com.sun.tools.javac, ...`
- Optimisations dynamiques
 - C (avec jeux de macros), ...



Techniques de génération dynamique de code

- *Modifications en place*
 - un programme (exécutable natif) se modifie lui-même : auto-adaptation au contexte d'exécution (OpenBSD n'autorise pas cette manipulation (W \wedge X))
- *Traduction de code-octet vers du code natif*
 - Problématique
 - perte (coût)
 - gain (avec amortissement du coût)
 - solutions (techniques d'implémentation)
 - Idées
 - traduction naïve (systématique)
 - traduction sélective (par instrumentation)
- *Autres usages de générations dynamiques de code*
 - *Implementing closures using runtime code generation*²
 - ...

²Martin Grabmüller, *Technische Universität Berlin*, auteur du langage *Turtle*

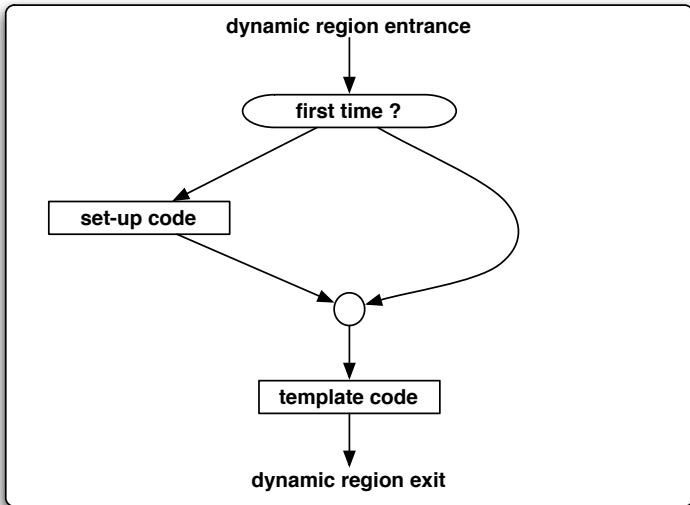
un exemple d'application de la compilation dynamique au langage C, avec modification dynamique

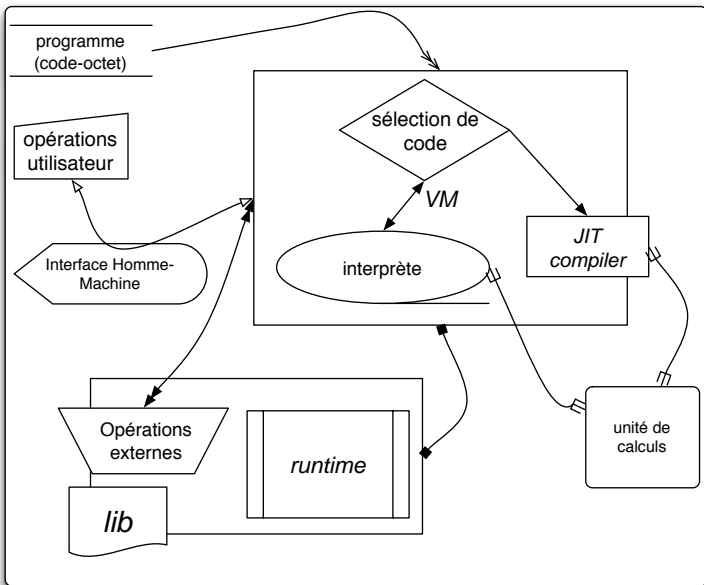
- utilisation des annotations (à l'aide d'un système de macros)
 - indiquer les zones à compiler dynamiquement
 - indiquer les *variables* qui sont *constantes* à l'exécution
 - indiquer les boucles à dérouler
 - ...
- génération de code à trous à l'aide de "modèles" (*templates*)
 - les trous sont remplis dynamiquement à l'exécution
- gains asymptotiques... (facteur 1,2 à 1,8)

Un petit exemple de code

```
cacheResult cacheLookup(void *addr, Cache *cache) {
    dynamicRegion(cache) { /* cache is runtime constant */
        unsigned blockSize = cache->blockSize;
        unsigned numLines = cache->numLines;
        unsigned tag = (unsigned) addr / (blockSize * numLines);
        unsigned line = ((unsigned) addr / blockSize) % numLines;
        setStructure **setArray = cache->lines[line]->sets;
        int assoc = cache->associativity;
        int set;
        unrolled for (set = 0; set < assoc; set++) {
            if (setArray[set]dynamic->tag == tag)
                return CacheHit;
        }
        return CacheMiss;
    } /* end of dynamic region */
}
```

Schéma de compilation dynamique à l'exécution





Un prototype pour l'optimisation dynamique par génération de code adapté aux fermetures

Travail de recherche (2006) de Martin Grabmüller
(Technische Universität Berlin)

- utiliser une autre représentation des fermetures (la représentation classique consiste en une structure de données)
 - perte : génération dynamique du code
 - gain : suppression d'une grande partie des indirections
 - idée : traitement des variables libres comme des constantes de la fonction

De nombreuses raisons de travailler sur la génération dynamique de code

- très adapté aux programmes réflexifs
- encore d'actualité malgré trois décennies de travaux
- nombreuses variantes
- nombreuses applications
- résultats (performances) pas toujours prévus
- méthodes de mise au point heuristiques demande beaucoup de temps (un peu comme pour les travaux sur les ramasse-miettes)