CST report

Théo Rogliano

June 5, 2020

1 General information

Doctorat : Informatique et applications Ecole Doctorale : Ecole doctorale Sciences Pour l'Ingénieur Lille Nord-de-France Etablissement : Université de Lille Date de la 1ere inscription en thèse : 1 octobre 2019 (1 A en 2019) Directeur de thèse : Stéphane DUCASSE Sujet de thèse :

2 Thesis subject and objectives

Sujet initiale: Noyaux de langage multiples. La thèse va évaluer et concevoir une solution pour avoir plusieurs noyaux de langages executés côte à côte.

Maintenant: il s'agit d'explorer les differents mécanismes d'isolations et les differents systèmes de communication entre processus. Le tout afin de créer un système semblable à un micro système avec plusieurs petits noyaux qui peuvent être remplacer au besoin (par exemple si un de ces derniers tombent en panne).

3 State of the art

4 Progression and results

4.1 Isolation

In order to transfrom the vm with one interpreter to multiple threaded interpreters, the goal has been to 'group' the global variables used by the interpreter and that each interpreter uses its own set of variables.

The first part was already (almost) done in the vm. All the globals were already grouped in a structure called foo (we call it interpreterState) and all access to those globals were done by using a macro GIV(var) foo-¿var. So, instead of having a lot of globals we have one global, the interpreterState structure. Note: there is some other globals still, i added some into the interpreterState (like the primitiveFunctionPointer)

The second part, that each interpreter uses its own interpreterState has been done in 3 main steps: -For boucle -Local thread variable -Argument passing The problem is we have a global that needs to be access by many threads.

For boucle: The interpreter being complex to read (low level, inlines) and touchy to change (one little mistake can rapidly break the whole thing) we started by modifying as little code as possible. The idea was to have an array of interpreterState of the size of the number of threads. Then each thread took one interpreterState by adding its ID in the interpreterState. To found back its interpreterState a thread had to go trough the array and found the interpreterState with its ID (a For boucle). Why is it simple to do ? We just need to add an array, add the for boucle in a function and change the macro to GIV(var) ForBoucleFunction()-¿var. Cons: it is slow as hell (10 times slower than the original interpreter).

Local thread variable: To speed things up we tought instead of having the interpreterState (or an array of interpreterState) as a global, the solution would be to have it in a variable local to the thread. The main difficulty here was that an interpreter uses already several threads (one for the interpreter, one for the heartbeat, one for the audio...) and all this threads should refer to the same 'local'. It takes to modify the code of this other components too. Cons: it is still slow (less than half the speed of the original interpreter).

Argument passing: The idea here is to add an extra argument directly at the creation of the thread which will contain the interpreterState. I omitted the fact that the code of the interpreter is generated. We modified the generation in order to add this extra argument first unused (already 30% slower than the original interpreter) and that's were i presented last benchmarks. Now to make this extra argument in used is more tricky. In the modified functions supporting the extra argument some of them are used in other files of the vm that are not generated (at least not by default). We can distinguish 2 cases, the plugins and the others (dunno how to call them). I started with the others, there are few and I modified them by hand. For the plugins, we started by thinking we could generate them but we did not find the generation for all of them and not everything were generated (for example in the FilePlugin, one file was generated but no the other). There is a lot of files to change (112) and a lot of different function calls to treat (1100). Since adding an extra argument is a refactoring, I tryied to handle this as a refactoring. In this order, I created an island grammar for the subset of C i needed to treat with PParser2. It allows to parse the plugins files(not perfect due to macro and some cases), create an AST for the subset of C and apply the refactoring with a visitor. In the end, after the refactoring there is still some compilation problem due to macro and the refactoring not being perfect. I corrected them and now the extra argument is on used everywhere and the impact is the same as when not in used (30 % slower than the original interpreter).

For the future, it could be nice to understand where the 30 % slow down comes from. I have a feeling that is something dumb and we can fix it easily. For example it could be the fact that i added the primitiveFunctionPointer in the interpreterState which is normal (each interpreter should ahve his own primitives) but complexify the primitiveFunctionPointer use.

4.2 Communication

Our goal through the channels is to enable communication between Process, OS Thread interpreters or another image. In our implementation, a channel is an atomicSharedQueue shared between Processes where one process can send objects or consume an object from the queue. In order to fully pass an object (and be thread safe) we added that the channel pass the ownership to one Process to the other in 2 different ways with: -A partial read barrier (Object¿¿become:). -A write barrier (Object¿¿beReadOnlyObject and Object¿¿beWritableObject)

Partial read barrier: we pass the ownership thanks to a become: when we send the object. the become remove all reference to the object from the sending process. Following the code of the send message:

send: anObject

— objectToSend — objectToSend := Object new. anObject become: objectToSend. queue nextPut: objectToSend

The process consumming the object in the channel gain a reference to the object and is now the only one with it. Pro: become: should be fast because become uses forwarder. Cons: some extra work are needed and should slow down the process if you want to give back the ownership

Write barrier: we start here by supposing all objects have an extra instance variable 'owner' which is the Process owning the object and are read only. Only the owner can write into IV of the object. When you send an object its owner become nil. When a process consume an object from the channel it become the owner. Pro: need less extra work than previous version. Cons: should be slow but (may be) could be optimize more. Some special objects like Array class or arrays do not support the become: message or to have an extra instance variable. So those objects cannot be send with a transfer of ownership with our implementation.

I think that will be a general problem for all classes with instances:

- if you change the ownership of a class, all its instances will point to the "revoked reference" =¿ so normal messages to the class should work =¿ but their instances do not! =¿ this is because this is a meta-object used by the VM

Future is to create example of application using channels and to compare both solution and enhance them if possible.

5 Plan for the next years

6 Formation followed

Catégorie : Formations numériques Gérer efficacement sa documentation avec Zotero - SPI (06 mars 2020 - 6 mars 2020) Crédit : 2 Total du nombre de crédits pour la catégorie Formations numériques : 2

Catégorie : Atelier technologique esug 2019 27th international Smalltalk Summer School (25 aout 2019 - 30 aout 2019) Groupe européen d'utilisateurs de Smalltalk,

Cologne - Allemagne 40 heures Crédit : 20 enregistrées par : Ecole doctorale Sciences Pour l'Ingénieur Lille Nord-de-France. Total du nombre d'heures pour la catégorie : 40 h Total du nombre de crédits pour la catégorie : 20

Total participation : 40 heures / 2 modules

Total des Crédits de Thèse : 22

7 Professionnal project

Je ne me vois pas continué dans en tant que post doctorant ou enseignant chercheur. Je ne suis pas non plus fan de l'instabilité des start ups, du moins le fonctionnement américain et doit me renseigner sur le fonctionnement français. Je me vois plutôt devenir ingénieur de recheche ou dans une entreprise.

8 Difficulties

L'épidémie de Covid a rendu le travail un peu plus difficile.