ANALYSE DE LA LECTURE DES FICHIERS XML

Jean-Marie Epitalon, Mars 2010

CERFACS, Working Notes WN-CMGC-10-20

Table des matières

1) Introduction

La lecture des fichiers XML par le driver d'OASIS4 semble prendre beaucoup de temps. Pour tenter d'améliorer l'efficacité de cette lecture, nous avons fait une étude sur le code source du logiciel OASIS4.

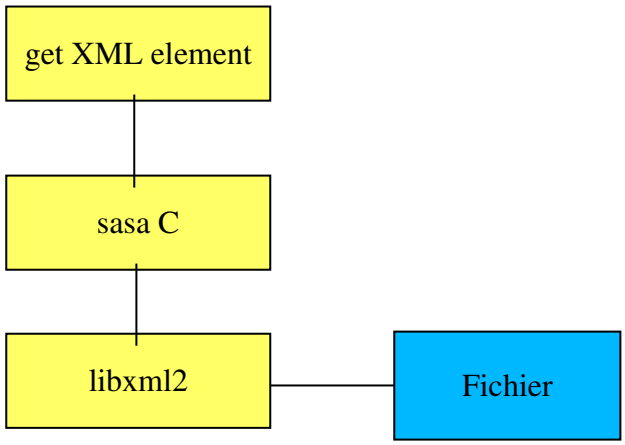
Ce document décrit les différentes étapes de l'étude et ses conclusions.

2) Architecture logicielle

Le logiciel OASIS4 utilise la bibliothèque open-source « libxml2 » pour accéder aux fichiers XML.

Il passe par une couche logicielle écrite en langage C, regroupée dans les fichiers sasa_c_xml.c et sasa_c_f90.c.

Au dessus de cette couche « sasa_c », une couche logicielle écrite en Fortran est chargée d'extraire le contenu d'un élément XML d'un fichier. Cette couche est constituée de tous les fichiers dont le nom est de type get_xxx.F90 présents dans l'unité logicielle « common ».



2.a) La couche sasa_c

2.a.1) Description

Le rôle de la couche sasa_c est double :

- servir d'interface entre le langage C de la bibliothèque libxml2 et le langage Fortran d'OASIS4
- fournir des services de haut niveau pour la lecture du contenu d'un élément XML ou d'un de ses attributs

En particulier, la couche sasa C met en mémoire tout le contenu d'un fichier XML, ce qui évite de relire ce fichier pour chaque élément d'information au moment où il est demandé.

2.a.2) Limitations et défauts

Après analyse, on a trouvé deux problèmes dans la conception de la couche sasa_c, ou plutôt, dans son utilisation (et donc dans la conception de l'architecture du logiciel) .

Dans la suite de ce document, nous présentons ces deux problèmes puis nous présentons la solution la plus simple pour chacun.

Limitation en nombre de fichiers XML

La couche sasa C est cependant limitée à un seul fichier XML à la fois. Cette limitation est gênante car le driver d'OASIS4 a besoin de lire les fichiers SMIOC des différentes composantes des applications,

Le driver d'OASIS4 ne lit pas les fichiers SMIOC l'un après l'autre mais chacun partie par partie. Il a donc besoin d'ouvrir plusieurs fichiers SMIOC simultanément, ce que la couche sasa C ne sait pas faire.

Par conséquent, le driver est obligé de demander à la couche sasa C de faire plusieurs fois la lecture et le décodage entier des mêmes fichiers XML. Ces répétitions obligées sont la cause principale de la lenteur du logiciel OASIS4.

On remarque que le driver d'OASIS4 lit chaque fichier SMIOC en 6 fois (4 fois avant l'ajout des grilles « user defined »). Voir en fin de document l'annexe qui analyse en détail les accès aux données des fichiers XML.

Imperfections internes

L'imperfection décrite ci-dessus vient d'une mauvaise conception de l'architecture : la couche sasa_c a été conçue pour un besoin limité alors que le logiciel OASIS4 a un besoin plus complexe.
La couche sasa_c a de plus quelques imperfections internes décrites dans ce chapitre.

L'extraction d'un élément XML passe par l'élaboration d'une chaine de caractère XPath,
Un exemple de chaine Xpath serait :

    //prismcomponent/transient/@2/intent

pour le sous-élément <intent> du deuxieme sous-élément <transient> d'un élément <prismcomponent> (composant d'une application).

Quand la couche « get XML » demande une information concernant un élément XML à la couche sasa_c, elle lui passe la chaine Xpath de cette information. Cette chaine Xpath est suffisante pour que la couche sasa_c retrouve l'information dans le contenu du fichier XML.

Il se trouve que la couche sasa_c reforme cette chaine Xpath en ajoutant un préfixe *defaut:* à chaque partie du Xpath. Ainsi, l'exemple donné ci-dessus devient :

        //defaut:prismcomponent/defaut:transient/@2/defaut:intent

Ceci prend du temps CPU et en plus, une erreur de programmation provoque, pendant cette transformation, une immobilisation inutile de mémoire jusqu'à la fin de l'exécution du programme.

Pour une explication technique sur les raisons de la transformation du Xpath, voir l'annexe II en fin du document.

3) Solutions proposées

3.a) Solution du problème No 1

Comme il n'est pas simple de modifier le fonctionnement du driver à cause de sa complexité, il vaut mieux modifier la couche sasa_c.

Il faudrait que la couche sasa_c soit capable de garder en mémoire le contenu de plusieurs fichiers XML. Ainsi, la couche sasa_c, sur ordre du driver, lirait et décoderait une fois seulement chaque fichier SMIOC.

La couche « Get XML element » irait ensuite demander les informations une par une à la couche sasa_c qui les extrairait de sa mémoire. Une extraction de la mémoire est nettement moins couteuse en temps qu'une extraction depuis un fichier.

3.b) Solution du problème No 2

La solution la plus simple est de modifier la couche « Get XML element » pour qu'elle produise directement une chaine Xpath avec le bon format : le format où tous les éléments sont préfixés par la

chaine de caractère « defaut ».

4) Annexe I : Occurence des lectures de fichiers XML

4.a) Lectures par le driver

Dans ce chapitre, on décrit la lecture des fichiers de configuration XML par le driver d'OASIS4. Le driver lit d'abord le fichier SCC.xml une fois puis il lit les fichiers SMIOC par parties.

4.a.1) Lecture du SCC : avant les SMIOCs

Graphe d'appel des procédures :

```
PRISMDrv_main
    --> PRISMDrv_Set_scc_info
        --> get_execution_mode
        --> get_dates
        --> get_... (plusieurs appels à des procédures get_xxx())
        -->
```

Le programme lit le SCC une seule fois. Ceci est fait dans la procédure get_execution_mode(). Les autres procédures de type get_xxx() vont chercher des informations mises en mémoire lors du get_execution_mode().
A la fin, le programme ne libère pas la mémoire allouée au moment du décodage du fichier XML.

4.a.2) Lecture des SMIOCs

Graphe d'appel des procédures :

```
PRISMDrv_main
    --> PRISMDrv_Set_smioc_info
        --> PRISMDrv_Init_smioc_struct
            --> PRISMDrv_get_udef_transients
                --> (Tous SMIOCs) get_smioc_transi_numbers
                --> (Tous SMIOCs)  get_transi_io_numbers
                --> (Tous SMIOCs)  get_transi_details
            --> (Tous SMIOCs)  get_smioc_numbers
            --> (Tous SMIOCs)  get_unitsets_details (pas appelée, dans les faits)
            --> (Tous SMIOCs)  get_grids_details
            --> (Tous SMIOCs)  get_transi_io_numbers
            --> (Tous SMIOCs)  get_persis_details (pas appelée, dans les faits)
```

Toutes les procédures dont le nom commence par « get_ » ouvrent, lisent puis ferment un fichier SMIOC. Chacune de ces procédures est appelée autant de fois qu'il y a de SMIOC.
Deux catégories de données semblent avoir existé dans les fichiers SMIOC des versions primitives d'OASIS4 et plus dans ceux de la version actuelle mais la lecture de ces données semble toujours être tentée par le programme :

- « Fortran unit sets » : lus par la procédure get_unitsets_details()
- « persistents » : lus par la procédure get_persis_details()

En fait, ces deux procédures ne sont pas appelées car elles sont appelées conditionnellement et la condition est toujours fausse . Mais leur code Fortran est toujours présent.

Au total, le driver OASIS lit et décode chaque SMIOC entièrement 6 fois. Il le faisait seulement 4 fois avant l'ajout des interpolations « user-defined »

4.b) Composantes des applications

Dans ce chapitre, on décrit la lecture des fichiers de configuration XML par les composantes des applications.

Une composante ne lit le fichier SMIOC qu'en mode « standalone ». Quand une application est couplée, elle reçoit sa configuration du driver.

Graphe d'appel des procédures :
PSMILe_smioc_init
--> get_smioc_numbers
--> get_unitsets_details
--> get_grids_details
--> get_transi_io_numbers
--> get_transi_details
--> get_persis_details

Comme pour le driver d'OASIS, les procédures get_unitsets_details() et get_persis_details() ne sont pas appelées. Au total, le fichier SMIOC est lu 4 fois.

5) Annexe II : Raison du reformatage des chaines XPath

Les éléments XML des fichiers SCC et SMIOC appartiennent à un domaine de nom ou *namespace*, associé à une URL : http://www.cerfacs.fr/PRISM/XML/1.1. Ceci provient de la mention xmlns="http://www.cerfacs.fr/PRISM/XML/1.1" qui apparaît en tête de tous les fichiers XML d'OASIS4.

Normalement, tout élément d'un namespace donné est préfixé avec un préfixe donné qui est propre à ce namespace. Dans notre cas, par exemple, le préfixe pourrait être « cerfacs ». Ainsi chaque élément XML, apparaitrait dans le fichier XML avec le préfixe cerfacs: on lirait par exemple

```
<cerfacs:transient local_name="SISUTESU">
    <cerfacs:transient_standard_name>surface_temperature</cerfacs:transient_standard_name>
```

Pourquoi n'est-ce pas le cas dans les fichiers SCC et SMIOC ?
Parce que le *namespace* associé à http://www.cerfacs.fr/PRISM/XML/1.1 est le *namespace* par défaut, du fait de la mention

xmlns="http://www.cerfacs.fr/PRISM/XML/1.1"

Pour que le préfixe « cerfacs » soit associé à cette URL, il faudrait qu'on mette la mention :

xmlns:**cerfacs**="http://www.cerfacs.fr/PRISM/XML/1.1"

à la place de l'autre en tête du document.

Dans notre cas, le nom de domaine est sous-entendu : c'est le nom de domaine par défaut. On n'a donc pas besoin de le mettre en préfixe à chaque élément XML.

Malheureusement, la bibliothèque libxml2 ne supporte pas les préfixes par défaut : c'est pourquoi on en ajoute un artificiellement avant de demander l'information.

Remarque : Il est préférable d'utiliser un domaine par défaut et de ne pas mettre de préfixe à chaque élément XML car cela alourdirait les documents XML. Cela serait différent si les fichiers XML contenaient une combinaison d'éléments XML venant de plusieurs horizons, le CERFACS entre autres.