Structures of distributed data and parallelism

Summary:

Contents

1 Qu'est-cequ' a structure of data distributed?

Currently, the code proposes two strategies to function in parallel via `MPI`:

- Pre and/or postprocessings using of elementary calculations (e.g. with the operator *Aster* `CALC_CHAMP`),
- Construction, assembly and resolution of the linear systems with the external products `MUMPS` or `PETSc` (e.g. `STAT_NON_LINE`). One solves here in parallel the usual total problem: one is in mono-field.

These two strategies of parallelism implement a distribution of the tasks and associated data. I.e. the processors will allocate the structures of usual data, them to initialize but they will fill them only partially. To facilitate the maintenance and the legibility of the code, one generally does not recut these structures of data to just. They thus comprise many zero values. To detect the complete or incomplete character of such `SD`, one of their attribute takes the value **`MPI_COMPLET`** or **`MPI_INCOMPLET`**.

Remarks :

For more information on these tools (usable also into sequential), one can consult the handbooks of Reference R 6.01.02/03 and R6.02.03, as well as the instruction manual U4.50.01.
To gain in performance, the code proposes two other complementary strategies of parallelism: in shared memory (`OpenMP` *) with the internal linear solvor* `MULT_FRONT` *(R6.02.02) and, with the tool* `ASTK` *who allows to launch independent studies.*

2 How is made the distribution?

Code_Aster being a code finite elements, the distribution of data put in work to control parallelism is a distribution by mesh (*'EOD'* for *Element Oriented Distribution*). Each processor is thus responsible for a package of meshs (physical or late) and will thus carry out only elementary calculations, the assemblies and the fillings of structures of data corresponding.

This distribution of the meshs between the processors is done on the level of the assignment of the model (order `AFFE_MODELE`). According to the mode of distribution chosen by the user (`MAIL_DISPERSE, MAIL_CONTIGU.`) and the number of processors allocated for the job, the code operates the distribution `MAILLE/PROC` and stores it in the structure of data `PARTITION`. This distribution can be modified in the course of calculation besides *via* the order *`MODI_MODELE`*.

Note:

Meshs tar divine: Meshs which do not exist in the initial grid and which are added during calculation to facilitate the setting in data. One generally meets them with the conditions of Dirichlet imposed with `AFFE_CHAR_MECA` or when the continuous method of contact is used
To facilitate the management of the late meshs those are automatically assigned with the main processor. This extra work generally induces weak a déséquilibrage of load. In the contrary case, the user can restore balance by discharging this processor (via the keyword `CHARGE_PROC0_MA/SD`) of a certain percentage of the physical meshs which are initially allotted to him (or of a certain number of under-fields if the distribution is directed under-fields).

3 What are they the structures of data concerned?

For the moment one "distributes" only them `SD` resulting from an elementary calculation (*`RESU_ELEM`* and *`CHAM_ELEM`*) or of a matric assembly (*`MATR_ASSE`*). Vectors (*`CHAM_NO`*) are not distributed because the induced profits report would be weak and, in addition, as they intervene in the evaluation of many algorithmic criteria, that would imply too many additional communications. The characterization *`MPI_COMPLET`* or *`_INCOMPLET`* intervenes in:

- `CELK (7)` for *`CHAM_ELEM`* (D4.06.05 – structure of Data field),
- `NOLI (3)` for *`RESU_ELEM`* (D4.06.05 – structure of Data field)
- *`REFA (11)` for `MATR_ASSE`* (D4.06.05 – structure of Data `sd_matr_asse`),

4 How does one handle such structures of data?

Since these structures of data preserve the same organization and same dimensions that their sequential version, one can handle them with the utility routines standards. However, for each total treatment (i.e. potentially implying all the meshs of the model), it is necessary to detect their complete or incomplete character (*via* flags of the preceding paragraph) and to adopt a strategy *ad hoc*.

For example, before a product matrix-vector one can supplement *`MATR_ASSE`* if necessary (*via `sdmpic`*) or to decide to stop in *`ERREUR_FATALE`* if this scenario corresponds to a advance unforeseen software. In other cases, one does not seek to supplement `SD` but the communications are established `MPI` corresponding to the total treatment concerned (e.g. search for maximum as in *`assmam`*).

Note:

To facilitate maintenance, the installation and the validation of the code, the calls are limited `MPI` with some utility routines: `mummpi`, `fetmpi` and `mpicm1` . If one wants to develop new communications, it is better thus to enrich one by these routines
Sequential version: One should in any rigour qualify them centralized. Because even a parallel calculation can lead to `MATR_ASSE` or of `RESU_ELEM` not distributed. It is enough for that the user chooses, for all calculation or right for a portion, the parallelism of the type `CENTRALIZE` in the operators `AFFE/MODI_MODELE` . Elementary calculations and the assemblies are then not paralleled, only the linear solvor is (`MUMPS` or `PETSc`).

5 Typical case (`MATR_DISTRIBUEE`)

1) When one carries out a calculation distributed with the solvor `MUMPS` , if the option is activated `MATR_DISTRIBUEE` , ends of `MATR_ASSE` clean with each processor are recut so as not to store of zero useless (`REFA (11)` is then taggé in `MATR_DISTR`) . But as a result the handling of these assembled matrices distributed "thinned down" is more complex. During certain total treatments (produced matrix-vector), one does not supplement them and one stops in `ERREUR_FATALE` . It will be necessary to enrich the perimeter by use of this functionality step by step.

Note:

To distinguish the data associated with each under-field, each one comprises one `SD_SOLVEUR` , one `NUME_DDL` , one `MATR_ASSE` and of `CHAM_NO` S adapted to its dimensions. In addition, each processor is responsible for one `SD_SOLVEUR` , of one `NUME_DDL` , of one `MATR_ASSE` and of `CHAM_NOs` Masters who will be quasi-vacuums and whose only function is to point towards `SD` slaves of the under-fields of which the processor with management. It is another form of `SD` distributed which thus implies this recursivity on two levels.

5.1 Rule to be respected during the programming of flood of data/parallel treatments

At the end of the operator *Aster* , all the total bases of the processors must be identical because it is not known if the operator who follows in the command file envisaged an incomplete flood of data input. It is thus necessary to organize, for example, the completeness of fields *ad hoc* in the routines of filing (while trying its to base on `MPI_ALLREDUCE` simpler and more effective to implement). To offer a more effective parallelism it will be necessary one day to make jump this bolt. But by then!