HUB, un Concentrateur de Ressources Distribuées en Combinatoire Algébrique

Christophe Carré* — *Université de Rouen, Département d'Informatique, Place Émile Blondel, 76130 Mont Saint-Aignan, France.* carre@dir.univ-rouen.fr

Sébastien Veigneau* — *Université de Marne-la-Vallée, Institut Gaspard Monge, 2, rue de la Butte Verte, 93166 Noisy-le-Grand Cedex, France.* Sebastien.Veigneau@univ-mlv.fr

Abstract

We present the project HUB. The aim is to produce an efficient environment to experiment in algebraic combinatorics. This tool, which relies on the ACE environment, allows to use programs outside from the computer algebra system *Maple*. It thus clusters inside a general system all the possibilities of specific and very efficient systems.

Résumé

Nous présentons les grandes lignes du projet HUB. L'objectif de ce projet est la production d'un outil d'expérimentation performant en combinatoire algébrique. Cet outil, qui s'appuie sur l'environnement ACE, permet d'utiliser des moyens de calcul extérieurs au système de calcul formel *Maple*. Il concentre ainsi au niveau d'un système généraliste les possibilités de systèmes spécialisés.

1 Introduction

L'essor actuel de la Combinatoire Algébrique provient d'une part de la possibilité d'expérimenter sur ordinateur et d'autre part du besoin croissant de méthodes effectives en mathématiques et dans les disciplines qui en font usage.

*Supported by Procope and the EC network "Algebraic Combinatorics".

D'un point de vue général, de nombreux systèmes de calcul sont désormais disponibles sur la plupart des machines du parc informatique mondial. Ce sont essentiellement des systèmes généralistes, c'est-à-dire offrant des possibilités de manipulation des objets mathématiques classiques en arithmétique, en analyse et en algèbre. Ces logiciels sont le plus souvent accompagnés de bibliothèques de fonctions relatives à des domaines plus spécialisés comme la théorie des groupes, la théorie des corps finis, l'algèbre linéaire, la géométrie, les statistiques ou la combinatoire. En pratique le calcul formel se concentre avant tout sur l'algorithmique de base sous-jacente au calcul mathématique dans sa grande diversité, touchant notamment l'arithmétique, l'intégration formelle, la résolution formelle des systèmes d'équations algébriques ou différentielles, mais aussi se penchant sur des problèmes comme la représentation efficace des données, souvent à la base d'une algorithmique performante. Le calcul formel fournit les outils de base fondamentaux à la mise en place de l'algorithmique mathématique.

Effectuer des calculs intensifs nécessite clairement une algorithmique sous-jacente optimale, le plus souvent disponible au cœur des systèmes dédiés, mais que l'on trouve tout aussi bien dans des systèmes généralistes. Pourtant, la performance n'est souvent au rendez-vous que lorsque l'algorithmique est associée à une programmation prenant en compte la nature et les spécificités des calculs. Cet investissement très lourd ne peut rentrer dans le cadre du développement de systèmes généralistes du fait même, nous venons de le voir, que les bibliothèques de fonctions sont des satellites et non des parties intégrantes du noyau. C'est là vraisemblablement une des différences primordiales entre les deux types de systèmes.

D'un autre côté, il existe plusieurs logiciels permettant d'effectuer des calculs en combinatoire algébrique. Très spécifiques, ils sont souvent performants dans leur domaine. Malheureusement, ces systèmes dédiés ont le défaut de leur qualité : spécialisés dans des domaines trop précis, ils ne laissent guère de possibilités en matière d'expérimentation. Car expérimenter, c'est parfois changer de point de vue et changer de point de vue c'est souvent changer de domaine. Ces logiciels ont tout de même la qualité de leur défaut puisque leur spécialisation garantit leur efficacité. Citons par exemple *SYMMETRICA*[1] développé principalement par A. Kohnert, de l'Université de Bayreuth. Le logiciel offre des outils en théorie des représentations, en théorie des invariants, sur les polynômes de Schubert, les fonctions symétriques et d'autres objets combinatoires. Citons également *SCHUR*[4], conçu par B.G. Wybourne, permettant de calculer des propriétés des groupes de Lie et des fonctions symétriques, ainsi que *GAP*[2] développé à l'Université de Aachen, en théorie des groupes.

L'utilisation de systèmes de calcul (notamment de systèmes généralistes) a apporté une dimension expérimentale nouvelle à la recherche en mathématique. L'obtention de nouveaux résultats est souvent liée à une démarche expérimentale qui conduit naturellement au développement de nouveaux algorithmes plus performants. Ces algorithmes sont alors intégrés au système même qui a contribué à leur élaboration. Mais quel est donc l'outil qui peut offrir de réelles possibilités d'expérimentation ?

Les logiciels de calcul formel se positionnent plutôt bien dans ce domaine. Ces systèmes généralistes et souples autorisent des changements de point de vue. Pourtant, en matière de performance ils ont bien du mal à rivaliser avec les programmes spécialisés.

Une solution consiste donc à développer des bibliothèques satellites de fonctions spécialisées, gravitant autour d'un moteur de calcul formel. C'est dans cet esprit que nous développons depuis 1994 l'environnement `ACE`[3], dédié au système *Maple*. La version 3.0 regroupe ainsi plusieurs bibliothèques permettant de manipuler les structures fondamentales de la combinatoire algébrique (partitions, compositions, permutations, mots, tableaux, différences divisées, fonctions symétriques) et de calculer dans les algèbres de groupe et les algèbres de Hecke des groupes symétriques et hyperoctaédraux. On y trouve aussi les caractères et les représentations de divers groupes classiques, ainsi que des objets de création plus récente tels que les polynômes de Schubert et les fonctions symétriques non commutatives.

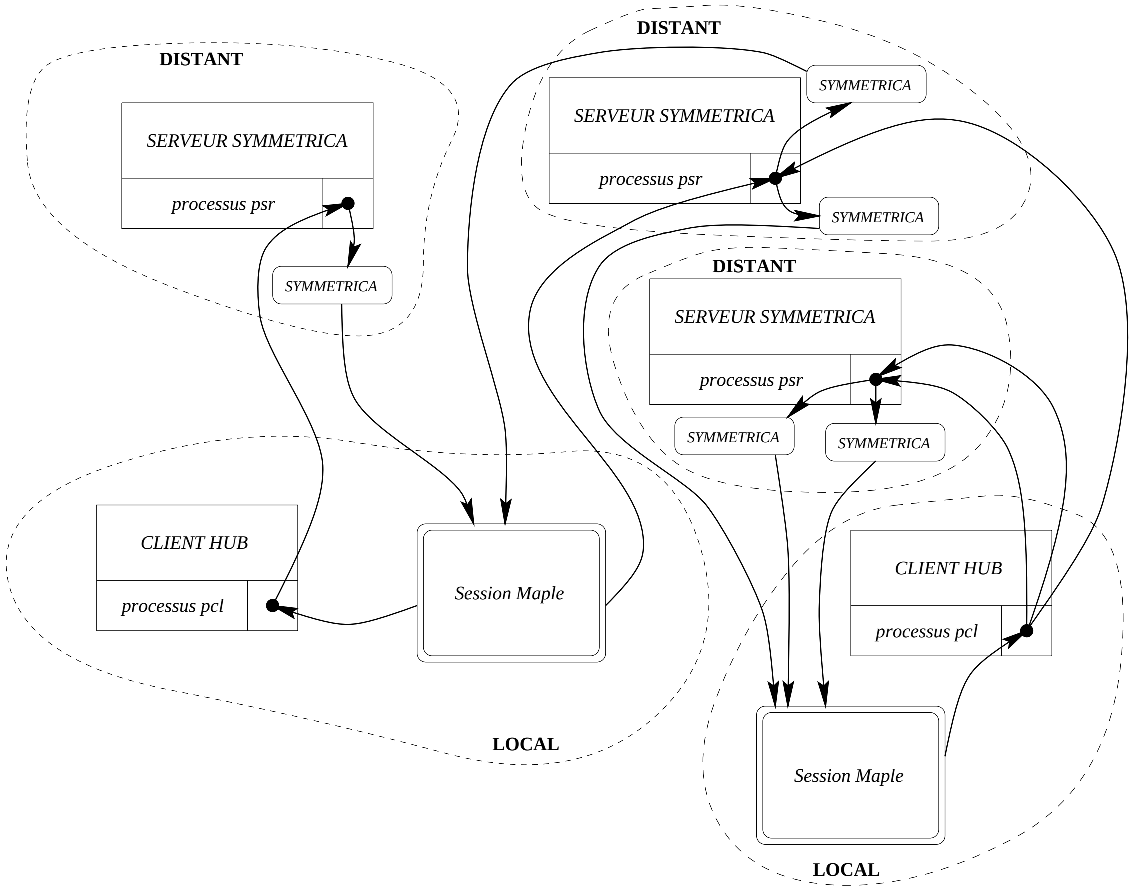
Et puis, nous avons parallèlement développé une surcouche de `ACE` permettant d'utiliser des moyens de calcul extérieurs à `ACE` et plus précisément extérieurs à *Maple*. Ce nouvel outil, que nous avons appelé `HUB`, permet de concentrer, au niveau de *Maple*, les possibilités de programmes extérieurs, programmes *C* ou plus généralement systèmes spécialisés comme *SYMMETRICA*. Notre objectif est simple : allier les possibilités d'expérimentation de *Maple* au potentiel de calcul des logiciels spécifiques. C'est ce point de vue que nous présentons maintenant.

2 Organisation du HUB

De manière générale, l'environnement `HUB` permet d'établir un lien entre `ACE` et des systèmes extérieurs à *Maple*. Par exemple, l'environnement charge `ACE` mais redéfinit certaines fonctions afin de les remplacer par des appels extérieurs à *Maple* réalisant les calculs plus rapidement. C'est ainsi que la table des caractères du groupe symétrique $\mathfrak{S}_n$ sera calculée par `ACE` jusqu'à $n = 9$ et calculée avantageusement par *SYMMETRICA* à partir de $n = 10$, les deux calculs étant lancés par la même commande. Par ailleurs, `HUB` met en place de nouvelles fonctionnalités dont `ACE` ne dispose pas. Par exemple, on pourra réaliser efficacement des calculs de pléthysmes de fonctions de Schur au niveau *Maple* en utilisant une fonction dédiée de *SYMMETRICA* plutôt que d'utiliser la fonction générique de `ACE`.

Globalement, le système `HUB` permet d'effectuer des calculs en utilisant de façon transparente des programmes lancés sur d'autres machines. Il se charge de détecter les calculs qui auraient intérêt à être exécutés hors de *Maple* et met en place un dialogue et des conversions de données entre les différents intervenants.

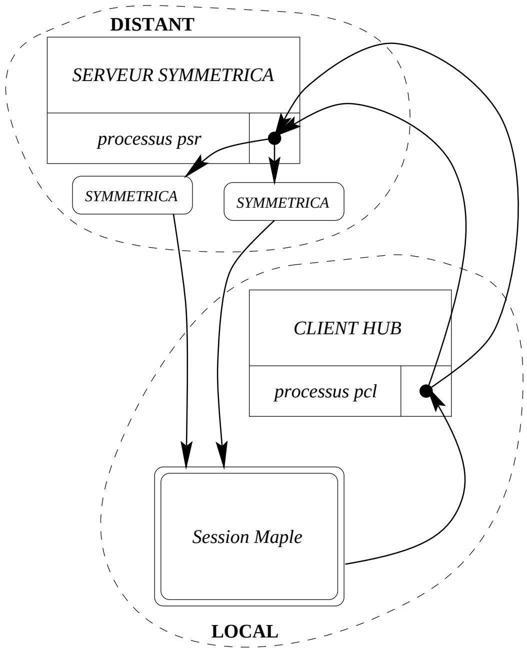
Plus précisément, prenons l'exemple de *SYMMETRICA* qui pourra se trouver sur une ou plusieurs machines distantes. D'un point de vue système, `HUB` est le client, *SYMMETRICA* le serveur et la communication se fait par *socket*. Chaque serveur *SYMMETRICA* possède un module de communication, écrit en langage *C*. Ce module, compilé puis exécuté, correspond à un processus que nous appelons "psr". Ce processus qui attend les demandes de connexion des clients, peut traiter parallèlement plusieurs connexions :



Chaque client `HUB` possède lui aussi un module de communication écrit en langage *C*. Une fois compilé, ce module correspond à un exécutable qui est appelé à chaque demande de service. Le processus “pcl” ainsi créé, essaye d'entrer en contact avec un (des) processus “psr”. Si le processus “psr” contacté n'existe pas ou n'existe plus (suite à une panne du système serveur par exemple. . .), ou si “pcl” constate que la connexion est trop lente ou encore si le processus “psr” traite déjà un nombre de connexions trop important, le processus “pcl” contacte un autre *SYMMETRICA*. A chaque demande de service, le processus “psr” crée un processus fils qui lit le service demandé sur sa *socket*. Ce nouveau processus exécute alors la commande *SYMMETRICA* appropriée et écrit le résultat sur sa *socket*.

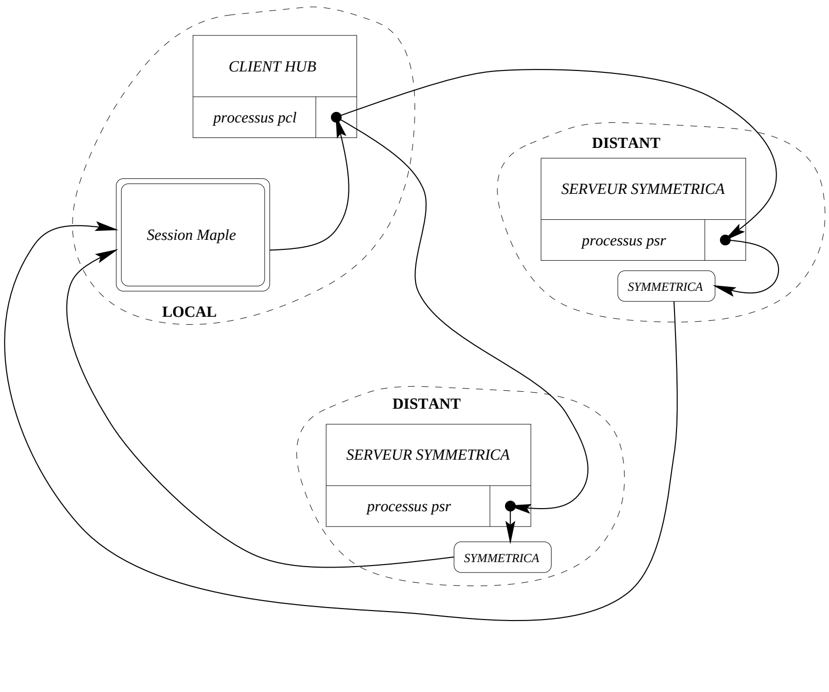
La communication entre *Maple* et l'extérieur se fait par l'intermédiaire de lectures et écritures dans les fichiers `/tmp/__HubOut.`<*pid de “pcl”*> qui contient les services demandés et `/tmp/__HubIn`<*pid de “pcl”*> qui stocke les réponses à ces services. Si la connexion est coupée en cours de calcul ou de transfert ou s'il y a un problème de lecture ou d'écriture au niveau des *sockets*, `HUB` en sera averti car le processus “pcl” retournera un code d'erreur.

Nous avons ajouté à l'environnement `HUB` des possibilités plus évoluées de distribution des calculs au sein même d'une session *Maple*. HUB peut ainsi demander que l'exécution de tâches extérieures s'effectue en arrière plan, ce qui correspond alors à une sorte de mode *background*. Nous avons mis en place deux méthodes pour arriver à ce type de schéma d'exécution. La première méthode fait que le processus client "pcl" crée n fils. Chaque fils essaye alors de rentrer en contact avec un serveur *SYMMETRICA*. Le processus "pcl" se terminera dès lors que les n fils auront fini leur calcul. Ce schéma correspond à la figure ci-contre dans laquelle "pcl" engendre deux requêtes aboutissant à la création de deux processus *SYMMETRICA*.



La deuxième possibilité permet de préciser qu'à la suite d'un appel extérieur l'environnement ne rentre pas en état d'attente du résultat.

Il poursuit alors d'autres calculs et une commande spéciale du HUB simule un point de synchronisation des différents calculs lancés en concurrence. Plus précisément, un utilisateur peut, au sein d'une procédure, lancer plusieurs calculs sur différentes machines distantes en même temps et reprendre ensuite en séquentiel l'exécution des instructions suivantes. Cette méthode est plus souple au sens où elle permet par exemple de mettre en concurrence un calcul *Maple* (en local) et un ou plusieurs calculs *SYMMETRICA* (distants). La figure suivante montre un exemple de distribution de deux calculs sur deux machines distantes :



Concrètement, le lancement de l'environnement HUB s'effectue à l'aide de la commande HUB qui initialise quelques variables tout en lançant *Maple* :

```
################################################################
# HUB environment                                 June 2, 1997 #
################################################################
# Christophe Carre           #      __/  __/ __/  __/ __/_/_/
# Departement d'Informatique #      __/  __/ __/  __/ __/   _/
# Universite de Rouen       #      __/_/__/ __/  __/ __/__/_/
# Place Emile Blondel      #      __/  __/ __/  __/ __/   __/
# 76130 Mont Saint-Aignan   #    __/  __/ __/_/__/ __/__/__/
#                            #
# Sebastien Veigneau         #
# Institut Gaspard Monge      #
# Universite de Marne-la-Vallee #
# 2 rue de la Butte Verte     #
# 93166 Noisy-le-Grand CEDEX  #          _/    _/_/_/
#                            #          _/    _/  _/
#############################          _/ _/ _/_/_/

HUB>
```

Maple peut alors être vu comme un fournisseur de ressources tant locales (grâce aux possibilités même de *Maple* augmenté de l'environnement ACE) que distantes. Les moyens de calculs distants sont représentés par une notion de service, qui peut tantôt être actif ou inactif. Pour continuer avec l'exemple précédent, si le service SYMMETRICA est inactif, les fonctions utilisant des ressources *SYMMETRICA* ne seront pas disponibles.

Par ailleurs, l'environnement HUB offre un certain nombre de services permettant de redéfinir les fonctions de ACE. Ainsi, l'activation du service SG redéfinira par exemple la fonction SgCharTable calculant la table des caractères d'un groupe symétrique donné. La fonction de ACE sera alors redéfinie en son homonyme dans HUB, cette dernière incluant des ordres de distribution du calcul pour des groupes symétriques de degré élevé. Maintenant, si le service SYMMETRICA est actif, le calcul de la table des caractères se fera effectivement hors de *Maple*. En d'autres termes, l'utilisateur peut contrôler à tout instant, en activant ou désactivant des services, si le calcul est fait en local ou à distance. Par défaut, tous les services sont actifs.

L'exemple suivant présente les temps de calcul de la table des caractères des groupes symétriques allant de $\mathfrak{S}_1$ à $\mathfrak{S}_{19}$:

```
HUB> for i from 1 to 19 do AceTime(SgCharTable, t.i, i) od:
HUB> print(seq(t.i, i=1..19));

     0, 0, .034, .066, .117, .250, .517, 1.083, 2.150, .600, .900,

     2.250, 3.017, 5.833, 10.800, 20.833, 37.484, 69.483, 137.083
```

Dans l'exemple précédent, le calcul de la table des caractères a été automatiquement réalisé par *SYMMETRICA* à partir de $n = 10$. Les résultats montrent qu'entre $n = 9$ et $n = 10$, le temps de calcul chute puis reprend sa croissance. En d'autres termes, le calcul distribué nous a permis de ramener le temps de calcul pour $\mathfrak{S}_{13}$ à celui correspondant à $\mathfrak{S}_{10}$ et ce avec exactement la même commande *Maple*. Le même calcul sans l'utilisation du service `SG` (qui fait appel au service `SYMMETRICA`) donne les temps suivants :

```
HUB> HubDisable(SG):
HUB> for i from 1 to 19 do AceTime(SgCharTable, t.i, i) od:
HUB> print(seq(t.i, i=1..19));

     0, 0, .016, .067, .117, .233, .517, 1.066, 2.200, 4.650, 8.750,

  17.300, 33.784, 69.716, 142.634, 298.666, 640.684, 1508.733, 3652.700
```

Ceci montre que l'utilisation d'un service spécialisé extérieur est très intéressante car elle va permettre de réaliser des calculs jusque-là impossibles ou trop longs au niveau du système *Maple*.

Ce n'est pas tout, car, nous l'avons déjà vu, l'environnement `HUB` propose de surcroît des primitives de mise en concurrence de calculs. De fait, les calculs seront plus rapides car bénéficiant des performances de logiciels spécifiques, mais en plus ils pourront avantageusement être lancés en parallèle sur plusieurs machines. C'est en ce sens que les perspectives seront intéressantes. Intéressantes à court terme par l'utilisation des logiciels existants mais aussi intéressantes à plus long terme en tirant profit des programmes et machines parallèles qui verront le jour.

Références

[1] A. Kerber, A. Kohnert and A. Lascoux, *SYMMETRICA, an object oriented computer algebra system for the symmetric group*, `http://btm2xd.mat.uni-bayreuth.de/axel/symneu_engl.html`, J. Symbolic Comp. **14** (1992), 195–203.

[2] M. Schönert and al, *GAP – Groups, Algorithms, and Programming*, Lehrstuhl D für Mathematik, RWTH Aachen, Germany, fourth ed., 1994.

[3] S. Veigneau, *ACE, an Algebraic Combinatorics Environment for the computer algebra system MAPLE*, *User's Reference Manual*, *Version 3.0*, `http://phalanstere.univ-mlv.fr/~ace`, Université de Marne-la-Vallée, 1997.

[4] B.G. Wybourne, *SCHUR, an interactive program for calculating properties of Lie groups*.