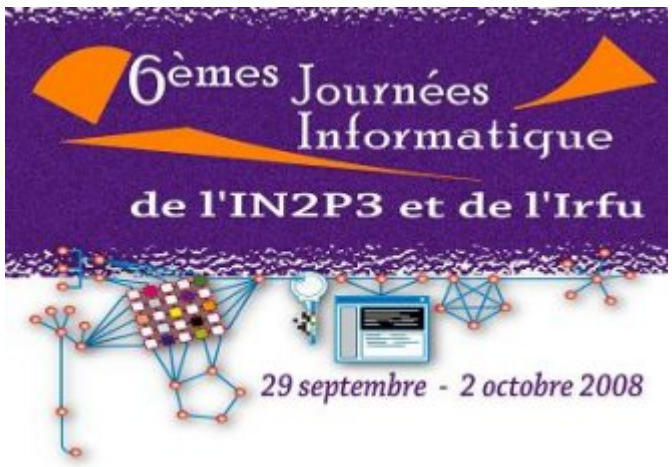
n°3
Novembre
2008

La lettre IN2P3 Informatique

Réseau des Informaticiens de l'IN2P3 et de l'IRFU

A LA UNE

Sixième édition des Journées Informatique de l'IN2P3 et de l'IRFU - Bilan



Les sixièmes journées informatique de l'IN2P3 et de l'Irfu se sont tenues du 29 septembre au 2 octobre derniers à Obernai en Alsace. Organisées tous les deux ans en moyenne, ces journées se veulent un lieu de rencontre de la communauté des informaticiens des deux instituts et l'un des temps forts du nouveau réseau thématique RI3. Nul doute cependant que d'une session à l'autre la formule évolue et évoluera encore.

Le succès de l'édition 2008 tient avant tout à la participation qui ne s'est pas démentie. Près de 135 participants étaient réunis à Obernai pour assister à ces journées mais ils n'étaient pas les seuls puisque la participation à distance rendue possible par le service de « webcast live » du Centre de Calcul a rencontré un certain succès. Si on en croit les premiers échos recueillis, ces JI08 ont été plutôt bien appréciées. La qualité des présentations est un ingrédient essentiel, mais le comité de programme a sans doute réussi la prouesse d'alterner les sujets sans multiplier les sessions parallèles sources de dilemmes, de ménager les temps de pause et d'oser quelques innovations comme l'organisation de tables-rondes et le lancement des JITV.

Un bon cru pour la sixième édition des journées informatique de l'IN2P3 et de l'Irfu

Fait assez remarquable en dépit d'un agenda chargé réparti sur 2 jours pleins et 2 demi-journées, (...)

lire la suite

■ Green Computing

Projet Ecoclim au LPSC Grenoble

Pour subvenir aux besoins de climatisation de ses salles informatiques le LPSC a mis en place un système original de refroidissement ayant les caractéristiques suivantes : Fonctionnement en free-cooling (85% du temps) Les flux d'air en face avant des machines sont séparés des flux en face arrière (principe du couloir froid) Refroidissement par simple échangeur air/eau utilisant l'eau industrielle (15% du temps) Ce système permet, outre une économie substantielle des consommations électriques, une moindre dépendance à la disponibilité de l'eau industrielle. Schéma Ecoclim La figure ci-dessus montre une vue de dessus de nos deux pièces informatiques. La pièce de droite contient les baies informatiques qui aspirent l'air en face avant (en bas sur le schéma) et le rejettent en face arrière (en haut sur le schéma). Noter que la face avant est séparée de la face arrière par une cloison pour éviter un recyclage intempestif. La pièce de gauche contient l'échangeur et le système de ventilation et de filtrage. L'air est aspiré à l'extérieur du bâtiment par la vanne d'arrivée d'air et soufflé à l'extérieur par la (...)

lire la suite

■ Science

1ère édition du Festival Particule.com

Les 20, 21 et 22 novembre prochain, l'Institut de Physique Nucléaire de Lyon (IPNL/UCBL-CNRS) et le Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules (CC-IN2P3/CNRS) organisent sur le domaine scientifique de la Doua (Villeurbanne 69), la 1ère édition du Festival Particule.com. Organisé à l'occasion de la Fête de la Science, le Festival Particule.com est un nouveau rendez-vous scientifique à destination du grand public et des scolaires. Une manifestation originale où théâtre se conjugue avec physique et informatique, pour placer les sciences au cœur de la culture. L'objectif de cette manifestation est en effet de sensibiliser le public (petits et grands) à l'informatique et à la physique sous un angle nouveau, éloigné des discours trop techniques que l'on entend souvent lors de la Fête de la Science. D'autre part, l'idée était de faire venir le public au sein même des lieux de recherche. C'est ainsi que le dôme de l'IPNL, un atelier technique d'ordinaire consacré au montage d'expériences, a été détourné pour en faire un lieu de (...)

lire la suite

■ Langage

Ada pour les nuls

Disons-le franchement, si vous vous considérez comme une nullité en programmation, Ada n'est pas fait pour vous... Ada est un langage qui enthousiasme ses utilisateurs, mais qui pourtant n'a pas su convaincre une grande partie des développeurs.C'est qu'il joue un rôle particulier dans le processus de développement: Ada a été conçu pour aider les développeurs à produire des logiciels de qualité industrielle, en renforçant les principes du génie logiciel. Ce qui n'implique pas qu'Ada soit déplaisant à utiliser, bien au contraire! Mais alors que la plupart des langages se vantent de la quantité de choses qu'ils vous permettent de faire, Ada met l'accent sur ce qu'il vous empêche de faire : des bêtises! Le compilateur assure, en plus de son rôle de traduction en langage machine, de véritables contrôles de cohérence qui ne laissent pas passer nombre de fautes de conception; c'est un véritable outil de génie logiciel, et non un simple traducteur. Ceci tend à rebuter les nouveaux venus qui ont l'impression que le compilateur les empêche de faire "ce qu'ils veulent". Ils ne perçoivent les avantages de cette approche que plus tard, car comme le (...)

lire la suite

■ Développement

NARVAL : Un système d'acquisition modulaire

NARVAL est un système d'acquisition hautement distribué développé en ADA et qui se sert des mécanismes offerts par ce langage pour toutes les communications inter processus. Ces mécanismes permettent de voir tous les processus constituant le système d'acquisition comme un programme unique et ainsi facilitent les communications entre chaque processus. Ils permmettent d'avoir une vision unitaire de l'ensemble du système. La gestion du flot de données entre deux processus est basée sur les protocoles TCP/IP et Infiniband lorsque les processus sont sur deux machines distinctes, ou par des fifo UNIX lorsque les deux processus sont sur la même machine. Le système d'acquisition utilise un nuage de processus pour gérer l'ensemble du flot de données. Chacun des processus hérite d'une classe abstraite nommée Acteurs définissant une interface commune. Il existe trois type d'acteurs : Producteurs : Acteurs dédiés à la collecte des données Intermédiaires : Acteurs permettant d'effectuer des traitements sur le flot de données ou servant de routeur NxM. Consommateurs : Acteurs (...)

lire la suite

■ Interview

Christian Helft : « Je rêve d'un lieu de rencontre encore plus authentique pour le RI3 »

Responsable du service informatique du LAL

lire l'interview

■ Archives

■ Abonnement

Pour vous abonner/désabonner, suivez ce lien.

■ Proposer un article

Vous souhaitez proposer un article ? Envoyez un mail à IN2P3Informatique@in2p3.fr.

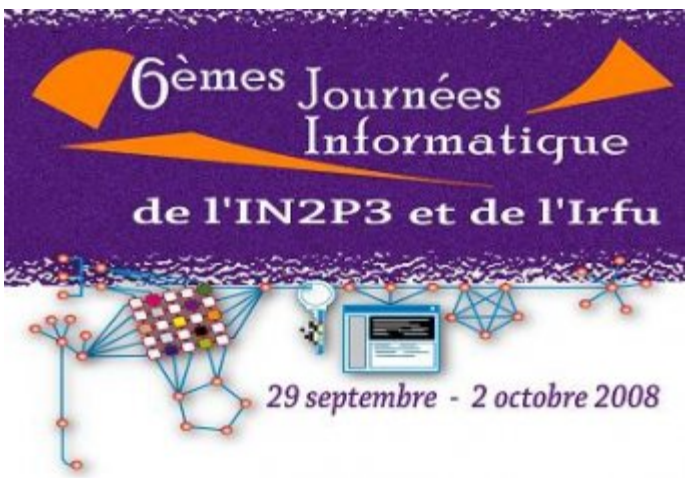
n°3
Novembre
2008

La lettre IN2P3 Informatique

Réseau des Informaticiens de l'IN2P3 et de l'IRFU

A LA UNE

Sixième édition des Journées Informatique de l'IN2P3 et de l'IRFU - Bilan



Les sixièmes journées informatique de l'IN2P3 et de l'Irfu se sont tenues du 29 septembre au 2 octobre derniers à Obernai en Alsace. Organisées tous les deux ans en moyenne, ces journées se veulent un lieu de rencontre de la communauté des informaticiens des deux instituts et l'un des temps forts du nouveau réseau thématique RI3. Nul doute cependant que d'une session à l'autre la formule évolue et évoluera encore.

Le succès de l'édition 2008 tient avant tout à la participation qui ne s'est pas démentie. Près de 135 participants étaient réunis à Obernai pour assister à ces journées mais ils n'étaient pas les seuls puisque la participation à distance rendue possible par le service de « webcast live » du Centre de Calcul a rencontré un certain succès. Si on en croit les premiers échos recueillis, ces JI08 ont été plutôt bien appréciées. La qualité des présentations est un ingrédient essentiel, mais le comité de programme a sans doute réussi la prouesse d'alterner les sujets sans multiplier les sessions parallèles sources de dilemmes, de ménager les temps de pause et d'oser quelques innovations comme l'organisation de tables-rondes et le lancement des JITV.

Un bon cru pour la sixième édition des journées informatique de l'IN2P3 et de l'Irfu

Fait assez remarquable en dépit d'un agenda chargé réparti sur 2 jours pleins et 2 demi-journées, l'ensemble des présentations a été suivi avec une presque étonnante assiduité. De nombreux aspects de la mise en œuvre et de l'utilisation de l'informatique ont été abordés. Mais le souci d'équilibre entre contributions invitées et présentations de « l'informatique IN2P3/Irfu », et le dosage entre les aspects Développement et ASR (Administration Systèmes Réseau) ont sans doute permis d'obtenir un assemblage de bonne tenue. « Je suis très content de cette instance des journées, qui a rendu compte d'une très large palette des préoccupations et des métiers de développeurs à l'IN2P3, ce qui n'avait pas toujours été le cas » constate Pierre-Yves Duval.

L'édition 2008 a reçu le soutien de l'IN2P3, du CEA, de RESINFO, IBM et DELL mais organiser les JI n'est pas une mince affaire ! Le comité d'organisation a assuré le suivi des opérations pendant presque une année. Une mention spéciale revient toutefois à l'équipe de l'IPHC qui a assuré avec brio, efficacité et gentillesse l'organisation locale de ce qui restera un très bon millésime alsacien.

En attendant la prochaine édition des JI, nous vous invitons à retrouver les archives vidéo des présentations ainsi que le « clip » et les interviews des JITV réalisés sur place sur le site ***http://indico.in2p3.fr/conferenceDi...***.

Conclusion vu par des intervenants

Rendre compte des temps forts de ces journées est une mission impossible que Geneviève Romier (Urec), Jean-Pierre Meyer (Irfu) et Bernard Boutherin ont acceptée et, on peut le dire, brillamment remplie.

« Tous les aspects du développement logiciel ont été abordés : des questions très proches du matériel comme la modélisation de cartes, la simulation ou l'émulation, la validation du matériel, les formats de données jusqu'aux aspects proches de l'utilisateur comme la visualisation des données, les cadriciels, l'authentification et l'autorisation » constate Geneviève Romier. « Ont également été évoqués les outils nécessaires au développement, les difficultés et exigences dans les collaborations, les défis technologiques auxquels l'informatique doit faire face dans ce domaine : temps d'exécution très contraints, performances pointues, souci avec la persistance et le volume des données, souci de la portabilité et l'optimisation des applications compte tenu de l'évolution des processeurs notamment. »

« Les JI sont l'occasion de ramener plein de choses à la maison » nous assure Bernard Boutherin. « Je garderai le souvenir d'une organisation au top. Je me suis enrichi d'un peu de culture générale comme les enjeux de la QCD sur réseau, ce qu'est réellement un cadriciel et une ontologie. Je ramène des idées qui peuvent servir tout de suite et des informations pratiques sur la gestion des configurations centralisées, l'intégration des services de grille dans l'exploitation de l'informatique du laboratoire, sur le service ORACLE au CC. Enfin je constate avec plaisir que le site des informaticiens ***http://informatique.in2p3.fr*** dont nous parlions depuis longtemps existe enfin ! Il y a également des tendances à surveiller, une évolution vers Windows qui offre beaucoup de possibilités mais qui fait débat et qui ne doit pas fermer la porte à une partie des utilisateurs. J'ai personnellement apprécié le concept des tables-rondes et je viens avant tout pour avoir des échanges conviviaux avec mes collègues. »

Jean-Pierre Meyer avoue quant à lui être l'un des rares physiciens à s'être glissé subrepticement dans ces journées. « J'ai beaucoup apprécié ces journées presque trop denses pour moi mais finalement je les ai suivies dans leur totalité. C'est un signe de l'intérêt que j'y ai trouvé. » « Si je dois résumer les points marquants, je suis avant tout déçu de devoir attendre encore 7 à 10 ans pour avoir une chance d'utiliser des processeurs 3D mais intéressé par la question de savoir si la physique des particules est réellement prête à exploiter les architectures multi-cœurs. Le parallélisme trivial (un job=un événement) nécessite de 2 à 4 Go par cœur. On n'échappera sans doute pas à une certaine forme de parallélisme de nos codes qui sont beaucoup trop gourmands en mémoire aujourd'hui. Un autre sujet de préoccupation concerne la pérennisation de la grille de calcul. Guy Wormser de l'institut des grilles du CNRS a évoqué les démarches en cours vers une infrastructure pérenne européenne EGI. J'ai noté avec beaucoup d'intérêt l'existence du site PLUME pour diffuser au sein du monde académique les logiciels issus de nos laboratoires mais je ne percevais pas les problèmes liés à l'utilisation de logiciels libres ».

Frédérique CHOLLET (LAPP) et Christian HELFT (LAL)

n°3
Novembre
2008

La lettre IN2P3 Informatique
Réseau des Informaticiens de l'IN2P3 et de l'IRFU

Christian Helft : « Je rêve d'un lieu de rencontre encore plus authentique pour le RI3 »

Responsable du service informatique du LAL

Cette année, et pour la première fois, [*1*] ont été réalisées aux JI. Comment vous est venue l'idée de ces clips vidéos ?

En consultant les archives de la conférence OGF23 (***www.ogf.org/OGF23***). Ils ont mis en ligne (en fait sur le site "GridCast, blogging behind the scenes of grid computing")(***http://gridtalk-project.blogspot.com***) toute une série d'interviews de participants qui donnent une autre "vision" de cette conférence que les traditionnels transparents. Comme nous étions en pleine préparation des JI08, j'ai soumis l'idée aux comités de programme et d'organisation, ce qui a soulevé un enthousiasme modéré je dois le dire…

Qu'attendez-vous de ces interviews ?

Je dois avouer que je suis un peu frustré par le format des JI. Je rêve d'un lieu de rencontre encore plus authentique des membres du réseau des informaticiens de l'IN2P3 et de l'IRFU. Et mon "rêve" va nettement au-delà d'une conférence traditionnelle. Comme je ne suis pas fanatique d'animation du style saut à l'élastique, j'ai pensé qu'au moins faire voir un peu l'envers du décor commencerait à "désenclaver" l'aspect rencontre. Et je trouve qu'Hélène (Kérec, au micro) et Gérard (Dreneau, à la caméra et au montage) ont fait très fort et ont vraiment répondu à mes attentes. Même si les questions d'Hélène sont parfois un peu déroutantes pour les interviewé(e)s, ce qui est finalement rigolo, je crois que ces clips montrent un côté des JI très complémentaire de celui présenté par le travail lui aussi remarquable (comme il nous y a habitués) d'Olivier Drevon au webcast et archives vidéos des présentations.

Mes autres idées pour faire évoluer les JI sont pêle-mêle de renforcer le côté workshop (fournir une sorte de moment fort aux groupes de travail, composantes continues de l'activité réseau de la communauté des informaticiens), d'avoir au moins une demi-journée de rencontre et dialogue entre fournisseurs de prestations (ASR et développeurs) et "clients" (les physiciens), d'établir des "salles de chat" qui permettrait l'échange de commentaires à chaud entre les participants sur tous les thèmes abordés pendant les journées, de permettre une authentique participation à distance pour ceux qui ne peuvent ou ne veulent pas se déplacer pendant plusieurs jours...

Comment avez-vous choisi les personnes à interviewer ?

J'ai laissé carte blanche à Gérard et Hélène, tout en leur indiquant quelques personnalités à ne pas manquer. Et à part les deux keynote speakers du premier jour et les deux piliers que sont Olivier Drevon et Nicolas Rudolf (l'"âme" du comité local d'organisation), tous les autres ont été vraiment choisis au hasard, ce qui était dans le cahier des charges...

Comment pensez-vous que ces vidéos doivent évoluer ?

Elles pourraient être encore plus diversifiées, allant du loufoque aux déclarations quasi institutionnelles. On pourrait aussi penser à en faire aussi "en libre service", dans le genre photomaton. Je me souviens d'une conférence "CHI" (Computer Human Interface) où cette forme avait connu un certain succès. Ou encore imaginer une question appelant une réponse très courte posée à un très grand nombre de participants avec un montage serré des réponses...

Propos recueillis par GAËLLE SHIFRIN

[*1*] des interviews filméesIRetrouvez toutes les vidéos des JI à l'adresse ***http://indico.in2p3.fr/conferenceDi...***.

n°3
Novembre
2008

La lettre IN2P3 Informatique

Réseau des Informaticiens de l'IN2P3 et de l'IRFU

Projet Ecoclim au LPSC Grenoble

Pour subvenir aux besoins de climatisation de ses salles informatiques le LPSC a mis en place un système original de refroidissement ayant les caractéristiques suivantes :

- Fonctionnement en free-cooling (85% du temps)
- Les flux d'air en face avant des machines sont séparés des flux en face arrière (principe du couloir froid)
- Refroidissement par simple échangeur air/eau utilisant l'eau industrielle (15% du temps)

Ce système permet, outre une économie substantielle des consommations électriques, une moindre dépendance à la disponibilité de l'eau industrielle.

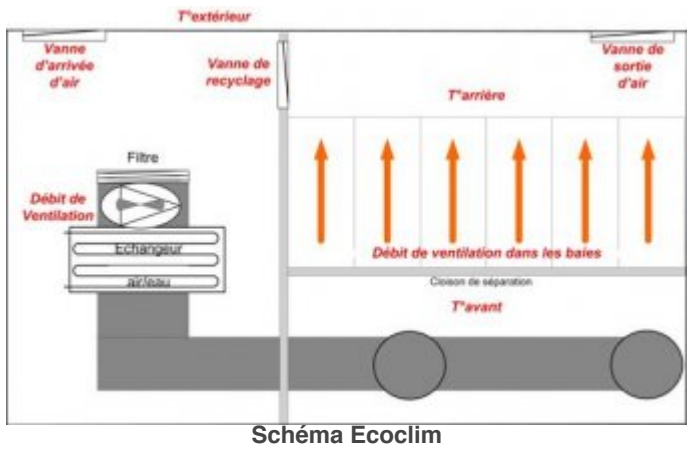
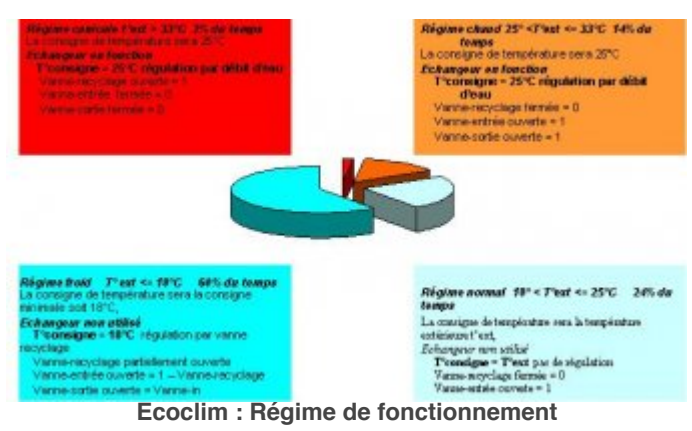


Schéma Ecoclim

La figure ci-dessus montre une vue de dessus de nos deux pièces informatiques. La pièce de droite contient les baies informatiques qui aspirent l'air en face avant (en bas sur le schéma) et le rejettent en face arrière (en haut sur le schéma). Noter que la face avant est séparée de la face arrière par une cloison pour éviter un recyclage intempestif. La pièce de gauche contient l'échangeur et le système de ventilation et de filtrage. L'air est aspiré à l'extérieur du bâtiment par la vanne d'arrivée d'air et soufflé à l'extérieur par la vanne de sortie d'air.

Régimes de fonctionnement

Le schéma ci-dessous montre les différents régimes de fonctionnement de notre système avec une température de consigne imposée en face avant.



Ecoclim : Régime de fonctionnement

Sur les conseils de notre installateur nous avons choisi des filtres G4 qui peut être lavé et remplacé au bout de 2 à 3 lavages, il est vendu en rouleau, pour un coût modique. L'hygrométrie de la salle informatique n'est pas contrôlée. En face avant des machines l'air peut donc être saturé en humidité ! Une simple récupération des condensats est faite sur l'échangeur. Etant donné que les machines chauffent il n'y a malgré tout pas de risque de condensation dans celles-ci.

Ces choix sont possibles aujourd'hui car

- Les serveurs supportent un environnement non parfaitement propre et une hygrométrie variable.
- Nous n'avons plus de bande magnétique en salle informatique grâce à la mise en place il y a deux ans de la sauvegarde déportée au Centre de Calcul de l'IN2P3 via le réseau RENATER.

Conclusion : Economie d'énergie et indépendance

Le premier intérêt de cette solution est l'économie d'énergie. En effet la puissance électrique de ventilation nécessaire que nous avons mesuré correspond à environ 8 à 9 % de la puissance installée au lieu de 33% de la puissance installée (COP de 3 pour une pompe à chaleur).

En mode free cooling - 84% du temps à Grenoble - on économise 75% de la puissance de refroidissement !

Notre installation est prévue pour une puissance de 80kW (nous avons 8 baies informatique 42U). A pleine puissance l'économie sera de 160 000 kWh/an ce qui permet de rembourser cette opération en 5 ans !

Le deuxième avantage de cette solution est d'être moins dépendant de la climatisation. En effet usuellement une panne de climatisation entraîne un arrêt de l'informatique. Dans notre cas ce type de problème n'a d'impact que si la température extérieure est supérieure à 30° soit moins de 5% du temps

Réduction de 95% des incidents informatiques dus à la climatisation !

Toutefois il est plus confortable de mettre la ventilation sur onduleur vu qu'elle ne représente que 8 à 9 % de la puissance installée.

Pour en savoir plus ***http://lpsc.in2p3.fr/informatique/2...***

Bernard BOUTHERIN (LPSC)

n°3
Novembre
2008

La lettre IN2P3 Informatique
Réseau des Informaticiens de l'IN2P3 et de l'IRFU

1ère édition du Festival Particule.com

Les 20, 21 et 22 novembre prochain, l'Institut de Physique Nucléaire de Lyon (IPNL/UCBL-CNRS) et le Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules (CC-IN2P3/CNRS) organisent sur le domaine scientifique de la Doua (Villeurbanne 69), la 1ère édition du Festival Particule.com.

Organisé à l'occasion de la Fête de la Science, le Festival Particule.com est un nouveau rendez-vous scientifique à destination du grand public et des scolaires. Une manifestation originale où théâtre se conjugue avec physique et informatique, pour placer les sciences au cœur de la culture. L'objectif de cette manifestation est en effet de sensibiliser le public (petits et grands) à l'informatique et à la physique sous un angle nouveau, éloigné des discours trop techniques que l'on entend souvent lors de la Fête de la Science. D'autre part, l'idée était de faire venir le public au sein même des lieux de recherche. C'est ainsi que le dôme de l'IPNL, un atelier technique d'ordinaire consacré au montage d'expériences, a été détourné pour en faire un lieu de représentation théâtrale. Au milieu du dôme trône aujourd'hui un véritable décor de théâtre créé pour l'occasion, décor représentant le LHC, le plus grand accélérateur de particules du monde, que le public pourra découvrir de l'intérieur en prenant la place des particules… tout un programme donc !

Les thématiques présentées dans le cadre du Festival Particule.com sont évidemment liées aux activités des deux unités : physique des particules donc, grilles de calcul, visite d'une salle informatique, etc. Pour l'occasion, un musée de l'informatique a également été installé au CC-IN2P3, avec l'aide notamment de l'Association ACONIT ***http://www.aconit.org*** (dont les missions sont la conservation et la restauration d'ordinateurs anciens) et une exposition retracera l'évolution du calcul de l'origine à nos jours.

Et pour représenter le lien (abstrait certainement pour les novices) existant entre la physique et l'informatique, pour rappeler que les expériences de physique ont un grand besoin de ressources informatiques, le circuit du Festival sera matérialisé par des fibres optiques, déroulées tout au long du parcours.

Le Festival Particule.com est réservé aux visites de scolaires les jeudi 20 et vendredi 21 novembre, et ouvert à tous publics le samedi 22 novembre, de 10 h à 18 h. N'hésitez pas à venir si vous êtes dans la région !

Site web de la manifestation : ***www.festivalparticule.com***

Gaëlle SHIFRIN (CC-IN2P3)

n°3
Novembre
2008

La lettre IN2P3 Informatique
Réseau des Informaticiens de l'IN2P3 et de l'IRFU

Ada pour les nuls

Disons-le franchement, si vous vous considérez comme une nullité en programmation, Ada n'est pas fait pour vous...

Ada est un langage qui enthousiasme ses utilisateurs, mais qui pourtant n'a pas su convaincre une grande partie des développeurs.C'est qu'il joue un rôle particulier dans le processus de développement: Ada a été conçu pour aider les développeurs à produire des logiciels de qualité industrielle, en renforçant les principes du génie logiciel. Ce qui n'implique pas qu'Ada soit déplaisant à utiliser, bien au contraire! Mais alors que la plupart des langages se vantent de la quantité de choses qu'ils vous permettent de faire, Ada met l'accent sur ce qu'il vous *empêche* de faire : des bêtises!
Le compilateur assure, en plus de son rôle de traduction en langage machine, de véritables contrôles de cohérence qui ne laissent pas passer nombre de fautes de conception; c'est un véritable outil de génie logiciel, et non un simple traducteur. Ceci tend à rebuter les nouveaux venus qui ont l'impression que le compilateur les empêche de faire "ce qu'ils veulent". Ils ne perçoivent les avantages de cette approche que plus tard, car comme le disent (de façon à peine exagérée) les utilisateurs convaincus: "en Ada, si ça compile, ça marche". Soyons honnête, il reste toujours des erreurs de logique inévitable, mais au moins, le langage vous protège de toutes les erreurs *évitables*.

Comment? C'est ce que nous allons essayer de montrer maintenant.

Un (bref) historique

Ada est un langage algorithmique d'une puissance d'expression considérable, dérivé de Pascal dont il a retenu les structures de contrôle et certains types de données. Il a été conçu suite à un appel d'offres du département américain de la Défense, d'après un cahier des charges très strict, par une équipe française dirigée par **Jean Ichbiah**. La première version du langage (Ada 83) synthétisait l'état de l'art de l'époque en fournissant un langage à la syntaxe lisible, mettant l'accent sur le typage fort, la modularisation, l'encapsulation des données, la réutilisabilité, la sécurité (statique, et dynamique par les exceptions) et la programmation parallèle et temps-réel.

Une révision majeure est intervenue en 1995 (Ada 95), qui a apporté l'héritage, de nouvelles possibilités de synchronisation, le support des systèmes distribués et une bibliothèque d'exécution plus étendue. La notion d'annexe a été introduite pour fournir des services complémentaires et optionnels pour certains domaines particuliers (programmation système, temps-réel, systèmes d'information...).

Enfin un amendement est intervenu en 2005 (Ada 2005), standardisé en fait début 2007, qui a remis le langage à la pointe des nouvelles technologies avec la notion d'interfaces, de nombreux perfectionnements pour la programmation temps-réel, et de nouvelles bibliothèques.

Ada est donc un langage bien vivant et qui suit l'évolution des technologies.

Syntaxe de base

Le programme ci-dessous est un premier exemple qui imprime "Bonjour", "Bon après-midi", ou "Bonsoir" en fonction de l'heure de la journée.

```
with Text_IO;  use Text_IO;
with Calendar; use Calendar;
procedure Bonjour is
 Heure : Day_Duration;
begin
 Heure := Seconds(Clock)/3600;
 if Heure < 12.00 then
 Put_Line("Bonjour");
 elsif Heure < 19.00 then
 Put_Line ("Bon après-midi");
 else
 Put_Line ("Bonsoir");
 end if;
end Bonjour;
```

On remarquera que toutes les instructions sont "fermées" par un **end**, ce qui rend inutile les regroupements par {...} que l'on rencontre partout dans les langages à base syntaxique C. Les clauses **with** et **use** en tête de la procédure permettent d'utiliser deux paquetages (*package*) qui fournissent respectivement l'accès aux fonctionnalités d'entrées-sorties et à l'utilisation du temps. Sous-programmes et paquetages constituent des unités

l'exception est variable: n'importe quel objet peut être "jeté" en C++, alors qu'en Java seuls les objets de la classe *throwable* peuvent l'être. Les exceptions Ada sont plus simples; ce sont de simples identifications, qui peuvent être "levées" en leur adjoignant éventuellement une simple chaîne de caractères. Ce choix a été motivé par la nécessité de conserver une bonne efficacité à ce mécanisme dans les contextes temps réel.

Héritage et programmation orientée objet

Comme C++, mais contrairement à Java et C#, Ada n'impose pas la programmation orientée objet, mais la fournit à ceux qui en ont besoin. Le modèle est original, et fondé sur la notion de type étiqueté (*tagged*). Ces types fournissent les fonctionnalités classiques de la POO:

- l'extension de types. Il est possible d'étendre le type de donnée en rajoutant des champs supplémentaires. Les opérations du type parent sont héritées et redéfinissables.
- Le polymorphisme. Des sous-programmes peuvent posséder des paramètres se référant à la classe d'un type étiquetté: ils sont applicables alors non seulement au type d'origine, mais également à tous les types qui en héritent.
- Les liaisons dynamiques. Si on appelle un sous-programme en lui fournissant un paramètre réel de type classe, le sous-programme effectivement appelé n'est pas connu à la compilation. C'est l'étiquette (*tag*) de l'objet réel qui détermine à l'exécution le sous-programme effectivement appelé.
- Les types abstraits. Il est possible de définir des types munis de l'attribut **abstract**, qui ne peuvent servir à définir des objets, mais seulement de racine d'arbres de dérivation pour des types concrets définis plus tard.
- Les interfaces. Ce sont des types abstraits spéciaux (sans données), qui fonctionnent comme les interfaces Java: un type étiqueté peut implémenter plusieurs interfaces simultanément. Ada offre de plus des interfaces *synchronisées* qui étendent la notion aux tâches et objets de synchronisation.

Remarquons que, contrairement aux autres langages orientés objets, la POO en Ada n'exige aucune allocation dynamique: il est donc possible d'effectuer des liaisons dynamiques sans imposer de pointeurs, qu'ils soient cachés (comme en Java) ou non.

Parallélisme

Le langage Ada permet de définir des tâches, qui sont des unités de programme s'exécutant en parallèle (comme les *threads* Java). C'est une fonctionnalité obligatoire, totalement intégrée au langage, et donc fournie par tous les compilateurs. Les tâches sont des objets appartenant à des *types tâche*, et peuvent donc être membres de structures de données: on peut ainsi définir des tableaux de tâches, des pointeurs sur des tâches, etc.

Deux moyens permettent aux tâches de se synchroniser et d'échanger des données: l'un est orienté traitement, c'est le *rendez-vous*; l'autre est orienté données, ce sont les *objets protégés*. Noter que ce sont des moyens de communications de haut niveau, beaucoup plus faciles à utiliser que les variables de condition de POSIX ou les méthodes *synchronized* de Java.

Dans le rendez-vous, une tâche exporte des services (appelés *entrées*) qui pourront être utilisés par d'autres tâches. Par exemple, un tampon simple se déclarera comme:

```
task Tampon is
 entry Lire (C : out Character);
 entry Ecrire(C : in Character);
end Tampon;
```

C'est la tâche elle même qui décidera (au moyen d'une instruction **accept**) quand elle est prête à prendre en compte la requète. C'est donc fondamentalement un modèle client-serveur.

Un objet protégé, en revanche est un type de donnée dont les opérations s'effectuent en exclusion mutuelle, comme dans des moniteurs de Hoare (ou des classes *synchronized* de Java). Mais les services fournis peuvent de plus être munis d'une condition, appelée *garde*, qui ne s'ouvre que lorsqu'une condition est remplie. On peut ainsi écrire de façon simple (et extrèmement efficace) des objets de synchronisation, comme la barrière simple suivante:

```
protected Barrière is
 entry Passer;
 procedure Ouvrir;
 function En_Attente return Natural;
private
 Ouverte : Boolean := False;
end Barrière;
```

de compilation, c'est à dire des structures pouvant être compilées séparément. Un programme complet est donc constitué d'un ensemble d'unités de compilation, fournissant ou utilisant des fonctionnalités aux autres unités. D'une façon générale, toute unité de compilation doit citer au début les autres unités qu'elle utilise au moyen d'une clause **with**: les dépendances entre unités sont toujours explicites, ce qui facilite grandement les analyses de dépendance dans les programmes.

Le mode de passage des paramètres de sous-programmes fait référence à *l'usage* qui en est fait, et non à la méthode de passage: les paramètres peuvent ainsi être déclarés **in** (lecture seulement), **out** (écriture seulement), ou **in out** (lecture et écriture). Ceci remplace avantageusement la notion de "passage par valeur" ou de "passage par référence" utilisée en C++. Par exemple:

```
procedure Decoder (Depuis : in String; Position : in out Positive; Valeur : out Float);
```

Cette procédure décode un nombre flottant depuis la chaîne "Depuis", à la position "Position" (qui est mise à jour en fonction des caractères lus), le résultat étant mis dans "Valeur". La lisibilité est encore améliorée en appelant cette chaîne en notation dite "nommée":

```
Decoder (Depuis => Chaine_Entree, Position => Index_Courant, Valeur => Resultat);
```

Typage fort

Un type définit un ensemble de valeurs muni d'un ensemble d'opérations portant sur ces valeurs. En fait, un type Ada représente des entités de plus haut niveau que les types d'autres langages: il représente une entité du domaine de problème, et non une entité machine. Le programmeur exprime les exigences de son problème; si par exemple il doit représenter des temps avec une précision absolue de 1 ms sur une année et des longueurs avec une précision relative de 10^{-5} jusqu'à 10^{25}m., il déclarera:

```
type TEMPS is delta 0.001 range 0.0 .. 86400.0*366;
type LONGUEUR is digits 5 range 0.0 .. 1.0E25;
```

Puisqu'il s'agit d'entités de nature différente, deux variables de types différents sont absolument incompatibles entre elles, même lorsqu'il s'agit de types numériques. Etant donné:

```
type Age   is range  0 .. 125; -- Soyons optimistes
type Etage is range -3 .. 10;
```

une variable de type "Age" est *incompatible* avec une variable de type "Etage". Autrement dit, en Ada, on ne peut pas "mélanger des choux et des carottes". Ada est le seul langage à offrir cette propriété, qui paraît tout à fait naturelle aux débutants en informatique... et pose parfois des problèmes aux programmeurs expérimentés.

Ada offre des structures de données très puissantes pour la gestion des tableaux et la paramétrisation des types de données, sans équivalent dans les autres langages. Leur description complète serait trop longue pour cet article, mais notons qu'ils permettent d'avoir des données de taille choisie à l'exécution, sans recourir à l'allocation dynamique (*malloc*), et tout en préservant la sécurité (aucun débordement de tableau n'est possible).

Modularisation

Une notion centrale en Ada est celle de paquetage (*package*). Le paquetage permet de regrouper dans une seule entité des types de données, des objets (variables ou constantes), et des sous-programmes (procédures et fonctions) manipulant ces objets. Le paquetage comporte une partie *visible* comportant les informations utilisables à l'extérieur du paquetage, et une partie *privée* (cachée) qui regroupe les détails d'implémentation auxquels les utilisateurs du paquetage n'ont pas accès.

Les paquetages peuvent être compilés séparément, et seule leur spécification est nécessaire pour permettre l'utilisation du paquetage. On réalise ainsi une séparation complète entre les spécifications d'une unité fonctionnelle et son implémentation. L'exemple ci-dessous montre la spécification d'un paquetage de gestion de nombres complexes:

```
package Nombres_complexes is
   type Complex is private;
   I : constant Complex;

   function "+" (X, Y : Complex) return Complex;
   function "-" (X, Y : Complex) return Complex;
   function "*" (X, Y : Complex) return Complex;
   function "/" (X, Y : Complex) return Complex;
   function Cmplx (X, Y : Float) return Complex;
   function Polar (X, Y : Float) return Complex;
private
   type Complex is
      record
         Reel : Float;
         Imag : Float;
      end record;
    I : constant Complex := (0.0, 1.0);
end Nombres_complexes;
```

Il est possible de créer des paquetages "enfant" qui rajoutent des fonctionnalités à des paquetages existant sans avoir à toucher à leurs "parents", donc sans recompilation des utilisateurs de ces parents. Ceci permet une meilleure séparation en fonctionnalités "principales" et fonctionnalités "annexes", facilitant l'utilisation aussi bien que la maintenance.

```
protected body Barrière is
 entry Passer is
 begin
 if Passer'Count = 0 then       -- Le dernier referme
 Ouverte := False; -- la barrière.
 end if;
 end Passer;

 procedure Ouvrir is
 begin
 if En_attente > 0 then
 Ouverte := True;
 end if;
 end Ouvrir;

 function En_Attente return Natural is
 begin
 return Passer'Count;
 end En_Attente;
end Barrière;
```

Ada supporte de nombreuses méthodes d'ordonnancement (priorités, EdF, budgets de temps...) qui peuvent même cohabiter dans une même application selon le niveau de priorité des tâches.

Gestion du bas niveau

Ada étant un langage également destiné à la programmation des systèmes, il fournit des fonctionnalités pour gérer le bas niveau. On peut ainsi de forcer la représentation physique des structures abstraites au bit près (de façon beaucoup plus précise que les *bitfields* de C++); par exemple, si une donnée sur 16 bits comporte un numéro d'identification sur 3 bits et une valeur sur 13 bits, on pourra les déclarer ainsi:

```
type Num_ID is range 0 .. 7;
type Valeur is range 0 .. 2**13-1;
type Donnee is
record
 Num : Num_Id;
 Val : Valeur;
 end record;

for Donnee use -- clause de représentation
record
 Num at 0 range 0 .. 2; -- Mot 0, bits 0 à 2
 Val at 0 range 3 .. 15; -- Mot 0, bits 3 à 15
 end record;
```

L'utilisateur pourra accéder au champ Num d'une variable D normalement, par la notation D.Num, sans se préoccuper de la représentation de bas niveau: c'est le compilateur qui se chargera d'extraire le champ de bits correct. On voit qu'ainsi, il n'est plus jamais besoin de calculer des masques, décalages, etc. pour intervenir à bas niveau. En cas de modification de la représentation interne des données, aucune modification du code n'est nécessaire, en dehors de l'ajustement de la clause de représentation.

Il est possible de spécifier des traitements d'interruptions physiques, d'inhiber des vérifications de type, et même d'inclure du code machine. Toutefois, des précautions ont été prises pour que même ce genre de manipulations ne puisse se faire sans un minimum de précaution et de sécurité.

Ada est également le seul langage qui définisse, au niveau de sa norme, les moyens de s'interfacer avec des bibliothèques écrites dans d'autres langages de programmation. Grâce à cette fonctionnalité, toutes les bibliothèques classiques (X-Window, Win32, GTK, Qt ...) sont disponibles pour Ada.

Distribution

L'annexe "systèmes distribués" définit certains éléments supplémentaires permettant d'écrire des systèmes distribués en Ada. Ada est le premier langage à inclure un tel modèle au niveau de la définition du langage, donc de façon portable et indépendante des systèmes d'exploitation ou des exécutifs sous-jacents (Java offre un modèle de distribution, mais il est lié au protocole Internet).

Un programme Ada est constitué d'un ensemble de "partitions". Les partitions peuvent être actives ou passives. Les paquetages constituant le programme sont classifiés de façon à assurer leur mise en œuvre automatique en environnement distribué; par exemple, les paquetages *remote call interface* offrent des services appelables à distance, sans que le programmeur ait à se préoccuper du mécanisme sous-jacent. Il est même possible de compiler le même programme, au choix, comme un seul exécutable pour une exécution locale, ou comme un ensemble de partitions pour une exécution distribuée.

Le modèle de distribution permet également de définir des objets (au sens de la POO) distribués, toujours de façon transparente pour le programmeur.

Le compilateur GNAT

Comment essayer Ada? Rien de plus simple. GNAT est une version du célèbre compilateur libre Gcc qui offre une implémentation complète du langage. Si cet article vous a donné envie d'essayer Ada (ou si vous avez des doutes sur la réalité des bonnes choses qui ont été présentées...), il suffit de le télécharger et d'essayer par vous-mêmes. Mais un bon conseil: lisez quelques-uns des guides ou des tutoriaux référencés ci-dessous. Le but d'Ada est de vous permettre de découvrir une

Génériques et réutilisation

Une autre propriété très importante du langage Ada est la notion d'unités génériques. Il s'agit d'unités paramétrisables, qui permettent de définir un algorithme indépendamment des types d'objet manipulés. L'idée des génériques a été par la suite reprise en C++ avec la notion de *template*. Notons cependant une différence importante avec C++: les génériques Ada sont compilés et vérifiés à la compilation; toutes les instanciations ultérieures fonctionneront forcément correctement (ou bien elles ne compileront pas si elles ne respectent pas le contrat demandé). Par exemple, le générique suivant permet d'obtenir une procédure qui permute ses arguments:

```
generic
 type Elem is private;
procedure Swap (X,Y : in out Elem);

procedure Swap(X,Y : in out Elem) is
 Temp : Elem;
begin
 Temp := X;
 X    := Y;
 Y    := Temp;
end Swap;
```

Exceptions

Les exceptions fournissent un moyen commode de traiter tout ce qui peut être considéré comme "anormal" ou "exceptionnel" dans le déroulement d'un programme. La plupart des langages récents fournissent la notion d'exception, mais la "granularité" de nouvelle approche du développement logiciel, plus sûre et plus efficace. Cela nécessite un petit effort pédagogique...

Références :

- *Ada sur Wikipedia :* ***http://fr.wikipedia.org /wiki/Ada_(l...***)
- *Wiki book Ada (français) :* ***http://fr.wikibooks.org /wiki/Progra...***
- *Wiki book Ada (anglais) :* ***http://en.wikibooks.org /wiki/Ada_Pr...***
- *Norme Ada :* ***http://www.adaic.org/standards/05rm...***

Ressources :

- *Compilateur GNAT :* ***http://gnuada.sourceforge.net/***
- *AdaPower :* ***http://www.adapower.com/***
- *AdaWorld :* ***http://www.adaworld.com/***

Groupes d'utilisateurs :

- *Ada France :* ***http://www.ada-france.org/***
- *Ada-Europe :* ***http://www.ada-europe.org/***
- *SIGAda :* ***http://www.sigada.org/***

Groupes de news :

- *En français :* news:fr.comp.lang.ada
- *En anglais :* news:comp.lang.ada

Jean-Pierre ROSEN (ADALOG)

n°3
Novembre
2008

La lettre IN2P3 Informatique

Réseau des Informaticiens de l'IN2P3 et de l'IRFU

NARVAL : Un système d'acquisition modulaire

NARVAL est un système d'acquisition hautement distribué développé en ADA et qui se sert des mécanismes offerts par ce langage pour toutes les communications inter processus. Ces mécanismes permettent de voir tous les processus constituant le système d'acquisition comme un programme unique et ainsi facilitent les communications entre chaque processus. Ils permmettent d'avoir une vision unitaire de l'ensemble du système. La gestion du flot de données entre deux processus est basée sur les protocoles TCP/IP et Infiniband lorsque les processus sont sur deux machines distinctes, ou par des fifo UNIX lorsque les deux processus sont sur la même machine.

Le système d'acquisition utilise un nuage de processus pour gérer l'ensemble du flot de données. Chacun des processus hérite d'une classe abstraite nommée Acteurs définissant une interface commune. Il existe trois type d'acteurs :

- Producteurs : Acteurs dédiés à la collecte des données
- Intermédiaires : Acteurs permettant d'effectuer des traitements sur le flot de données ou servant de routeur NxM.
- Consommateurs : Acteurs utilisés en fin de chaine afin de stocker ou de visualiser les données.

Rapide historique

Initialement, NARVAL a été développé à l'IPN Orsay pour remplacer OASIS, l'ancien système d'acquisition de l'IPN Orsay. De système d'acquisition interne à l'IPN Orsay, l'utilisation de NARVAL s'est étendu à d'autres laboratoires de l'IN2P3 et du CEA au fil des années et est, ou a été, utilisé dans des expériences au GANIL, à Los Alamos, ...

Depuis 3 ans, l'IPN Orsay et le CSNSM travail de concert à l'amélioration de cet outil. Plus récemment, le GANIL s'est intéressé à ce développement et effectue des tests afin de savoir si NARVAL peut servir de flux principal d'acquisition pour une future acquisition au GANIL.

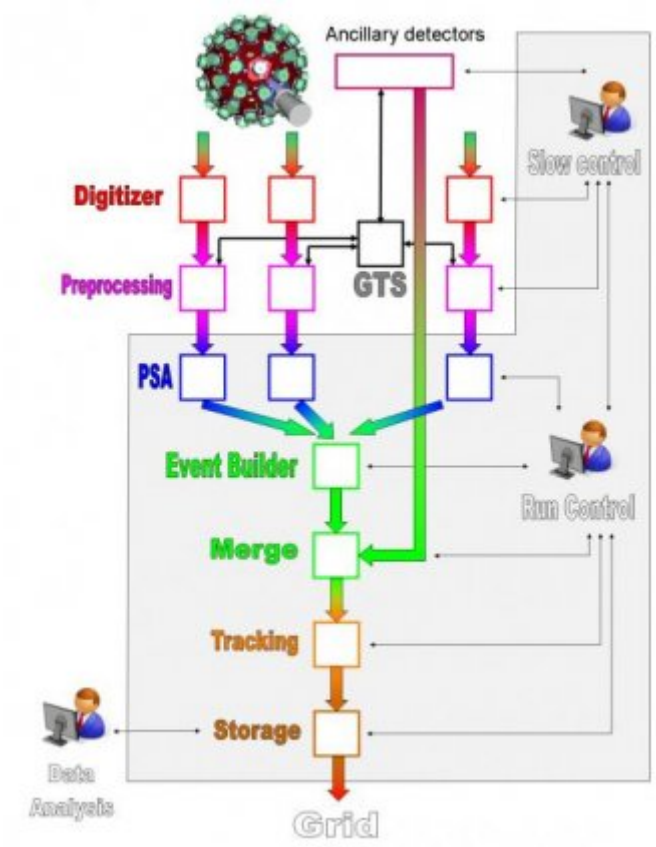
Concepts

Le développement de NARVAL a été conditionné par 4 impératifs : la modularité, la fiabilité, la réutilisabilité et l'interconnexion avec d'autres programmes.

Chaque acteur hérite d'un ensemble de fonctionnalités définissant les protocoles de synchronisation entre acteurs, la structure du nuage d'acteurs, ... Ainsi, sans aucun ajout de code, un acteur est capable de communiquer avec l'ensemble du système d'acquisition, de recevoir les données des acteurs lui en fournissant et d'envoyer des données aux acteurs en attendant. La seule différence entre deux acteurs est le code métier embarqué.

Cette possibilité de créer facilement des acteurs permet de les spécialiser. Il existe des acteurs dédiés à l'acquisition de données, ayant la fonction de filtre, de constructeur d'évènements ... Le fait d'avoir spécialisé les acteurs et de pouvoir facilement multiplier le nombre d'acteurs, autorise à créer des acteurs ne remplissant qu'une seule tâche bien définie. Cela a une grande importance pour tout ce qui concerne le test et la correction d'erreurs. Effectivement, une fois l'erreur détectée, il est aisé d'isoler l'acteur erroné, puis de le corriger.

De la même manière, cette possibilité d'isoler chaque traitement, permet de simplifier la validation d'un nouveau traitement. Comme tous les acteurs possèdent le même code pour tout ce qui concerne la gestion du flot de données, ces fonctionnalités ont déjà été validés, et donc il ne reste plus qu'à tester le code spécifique. Enfin NARVAL est développé en ADA, langage doté d'un typage très fort et d'un compilateur strict, ce qui permet de s'affranchir assez simplement de beaucoup d'erreurs syntaxiques et sémantiques.



Architecture pour l'expérience AGATA

Enfin, grâce aux développements effectués pour le projet AGATA, NARVAL fournit des acteurs génériques. La norme ADA fournit un ensemble de règle pour lier du code C/C++ avec du code ADA. Nous nous sommes servis de cette définition, pour développer des acteurs contenant toute la "logistique" des acteurs standards (communication, transfert de données, synchronisations, ...), mais dont tout le code métier est géré par une librairie dynamique. L'interface de cette librairie est le résultat du travail de X. Grave (IPN Orsay) et V. Pucknell (Daresbury).Il est donc tout à fait possible de développer une librairie C/C++ et de se servir de NARVAL pour transporter les données.

NARVAL par l'exemple

Pour reprendre l'exemple du projet AGATA, NARVAL prend les données en sortie de cartes d'électroniques, effectue cinq types de traitements différents sur le flot de données, avant de stocker ces données. Lorsque le détecteur AGATA sera complètement opérationnel, NARVAL acquerra les données par un ensemble de 180 acteurs. Ces 180 flots de données parallèles seront traités par une première série d'acteurs dédiés au Pulse Shape Analysis, ensuite ces flots de données seront assemblés de manière à ne créer plus qu'un seul flot consistant. Ce flot unique transite par plusieurs acteurs et est enfin stocké.

Chaque acteur fournit un travail très spécifique, et a été développé dans diverses équipes européennes. La majorité des algorithmes de traitement des données ont été développés en C++ en se servant d'une librairie partagée pour effectuer tous les traitements spécifiques.

Eric LEGAY (CSNSM)