Le C++ est-il adapté au calcul scientifique et à l'analyse de données ?

R. Ansari

LAL, 21 Juin 2005

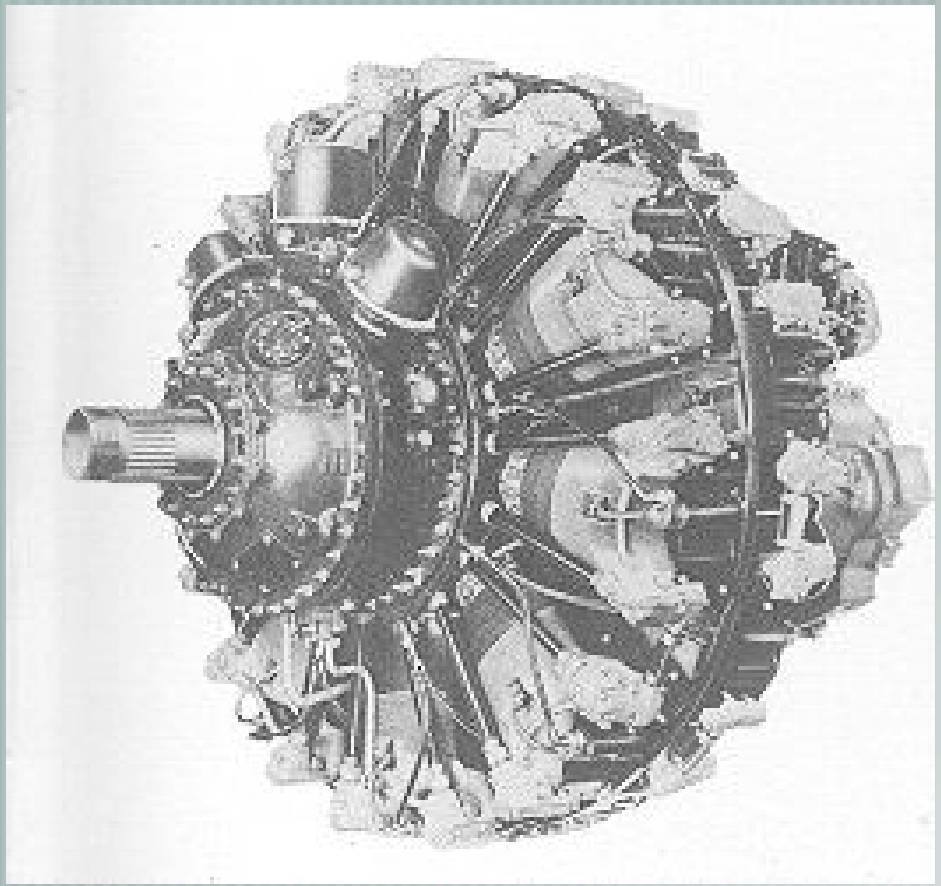
- La difficulté des évolutions technologiques
- Les langages objet
- Le développement de PEIDA++ pour EROS
- SOPHYA et ses choix techniques et architecturales
- Comparaison de performances
- Outils associés à SOPHYA

Le transport aérien dans les années 1950

Lockheed Constellation
1939-1958

Douglas DC 7
1953-1958

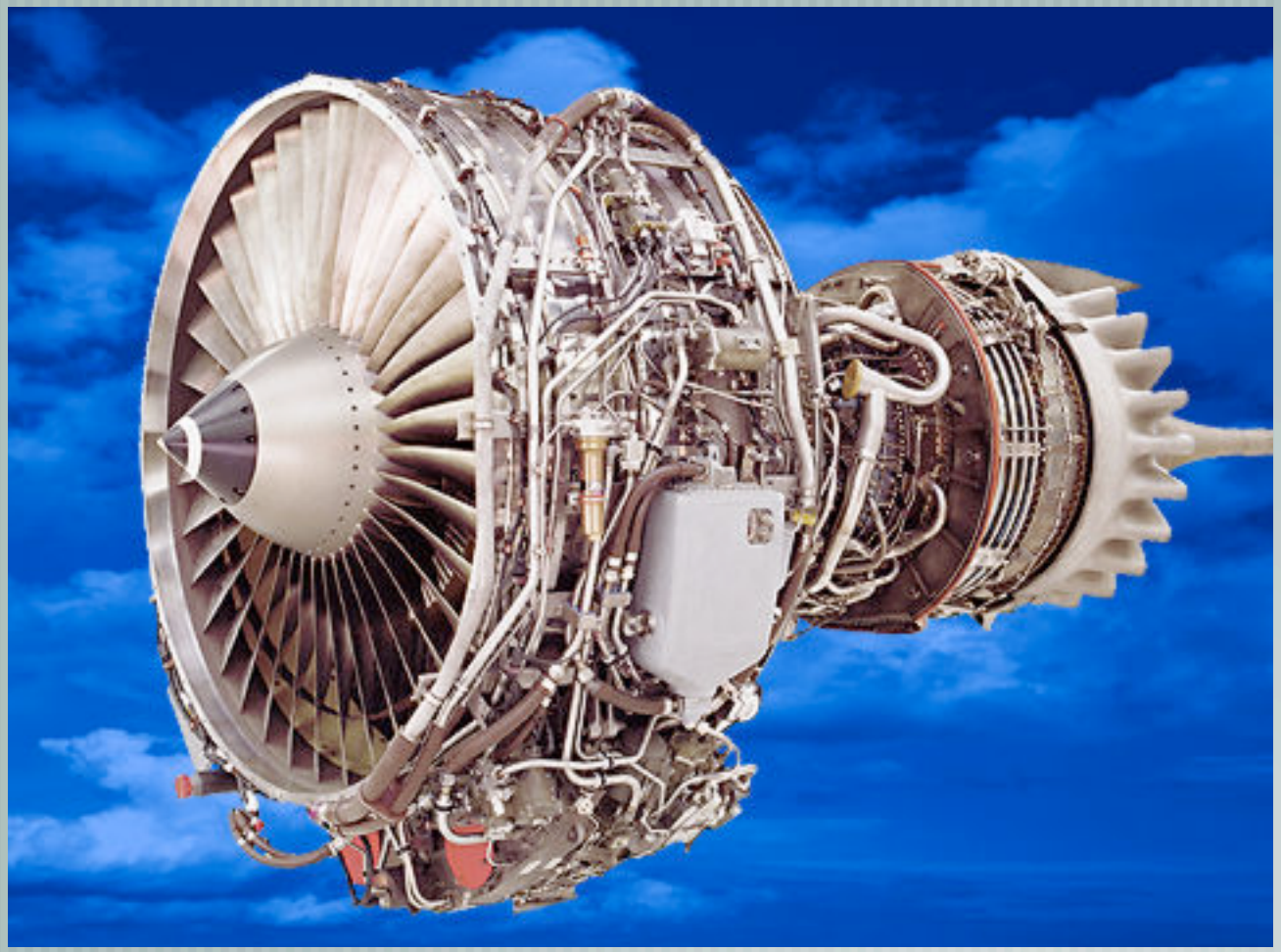
Évolution de la motorisation des avions de transport dans les années 1950



Pratt & Withney R-2800

- 18 cylindres en étoile
- 45.9 litres de cylindrée
- Poids: 1070 kg
- Puissance: 2500 chevaux
- 1939 ~ 1960

Turboréacteurs et turbo-propulseurs ...



CFM56 - 5C

- Poussée:140-150 kN
- Poids : 4000 kg
- Dilution: 6.5
- 1993- (A340)

Gestion de l'évolution technologique

De Havilland Comet 1 (Gde Bretagne, 1949)
premier avion de ligne à réaction
S'est désintégré deux fois en vol (1952, 1953)

Boeing 707

- Long courrier à réaction
- Premier vol : 1957
- Retour de Comet 4 en 1958

- Douglas Aircraft / McDonnelDouglas n'existe plus (Fusion avec Boeing)
- De Havilland n'existe plus (Fusion avec Hawker Siddely)
- Règne de Boeing sur le transport aérien pendant près de 30 ans
- Années 2000: Le combat des géants Airbus/Boeing

Quelques langages de programmation Une introduction historique

- Fortran : Premier (?) langage de “haut niveau” (**For**mula **Tran**slator)
  - Apparu en 1954/1955 (IBM)
  - Fortran IV en 1962
  - Fortran 77
  - Fortran 90/95

Programmation structuré , calcul d’expressions numériques

Le langage C

- Créé par D. Ritchie (AT & T / Bell Labs) pour Unix, sur PDP-11 (1972)
- Langage de programmation des systèmes d'exploitation (Unix en particulier)
- Normalisation ANSI/ISO C89 (1989)
- Nouvelle norme , C99 (1999)

Structures, pointeurs, allocation dynamique ...

Les langages orientés objet

- Technique introduite il y a 35 ans environ
- Simula (Norvège, 1967) - extension objet de Algol60
- Smalltalk (Xerox, 1970-1980)
  - Objets
  - Classes, instances
  - Messages, méthodes

Le C++

- Extension objet du C (C-avec-classe) par B. Stroustrup au début des années 1980's (AT&T)
- Cahier des charges:
  - Maintenir la compatibilité avec le C
  - Performance
  - Insertion dans l'environnement standard Unix
- Normalisation ISO/IEC 14882: 1998 *on the programming language C++*

Java

- Plateforme et langage Java élaboré par Sun au début des années 1990, utilisé au départ comme environnement de programmation objet embarqué (Oak/green OS 1992-1993)
- 1994-1996 : Utilisation pour des développements liés au Web/ internet ➟Java
- Années 2000 : Java devient un environnement de programmation largement utilisé

Java : caractéristiques

- Basé sur le C et le C++ (syntaxe proche)
- Simplification du C++ (pas de pointeurs, pas d'héritage multiple, pas de template ...)
- Gestion simplifié/automatisé de la mémoire
- Environnement complet de programmation, le code compilé Java tourne dans une machine virtuelle (JVM)
- Très riche bibliothèque de classes
- Chargement et lien dynamique

Autre langages

- Objective-C
- Eiffel (Voir Conception et Programmation par Objets, B. Meyer (1990))
- Ada 95
- Python

Environnement de programmation

- Librairies de support et librairies spécialisées
  - Exemple: librairie mathématique, CERNLIB ...
- Outils de développement (compilateur/éditeur de lien, outil de déverminage, IDE ...)
- Outils de gestion du code source et de déploiement (cvs, make, cmt ...)
- Outils d'analyse interactive
- L'interface avec l'infrastructure de calcul et la gestion de données (Base de données, Batch, Grilles, OS ...)

Environnement intégré de calcul technique et scientifique

- Langage interprété et environnement intégré/graphique de calcul technique, orienté tableaux et matrices
  - Matlab (http://www.mathworks.com/)
  - Scilab (http://scilabsoft.inria.fr/)
  - IDL (http://www.rsinc.com/)
- Système de calcul symbolique
  - Mathematica (http://www.wolfram.com/)
  - Maple (http://www.maplesoft.com/)

Comparaison C++/Java/F90 (1)

	C++	Java	F90
Structures	✓	✓	✓
Classes	✓	✓	
Polymorphisme statique	✓		
Polymorphisme	✓	✓	
Pointeurs	✓		✓
Références	✓	✓	

Comparaison C++/Java/F90 (2)

	C++	Java	F90
Allocation dynamique	✓	✓	✓
Création d'objets sur la pile	✓		
Gestion mémoire automatisée		✓	
Exceptions	✓	✓	
Module/Package/Namespace	✓	✓	✓
RTTI	✓	✓	

Comparaison C++/Java/F90 (3)

	C++	Java	F90
Opérateur de conversion	✓		
Surcharge d'opérateurs	✓		✓
Tableaux multi-dim/sous tableaux (natif)			✓
Programmation générique (template)	✓	1.5 ?	

- Le C++ est un langage puissant, mais difficile à maîtriser
- Ambiguïté de syntaxe
- La bibliothèque standard est puissant, mais a une couverture partielle (stdc++)
- Mais l'utilisation des bibliothèques existantes codées dans les langages de plus bas niveau (C, Fortran, ...) est aisé

EROS et PEIDA++

- PEIDA++ : Librairie de classes C++ à la base des programmes de reconstruction et d'analyse d'EROS
- Développé pour EROS 2, entre 1994-1998
- Utilisé pour les différents programmes scientifiques, dans les contextes en-ligne (Chili) et hors-ligne (CC-IN2P3,LAL,DAPNIA)
- Contributions de nombreux collègues (LAL, DAPNIA)

  E. Aubourg, D. Hardin, E. Lesquoy, C. Magneville, O. Perdereau, N. Palanque, N. Regnault ...

http://www.lal.in2p3.fr/recherche/eros/PeidaDoc/

PEIDA++

- Classes de représentation pour des objets d'usage général
  - images, matrices, vecteurs, histos ...
- et d'objets spécifiques
  - étoiles, liste d'étoiles, transformation de calage ...
- Mécanisme de persistance rudimentaire
  - fonctionnement sur des flots, mais le contenu et la structure étaient uniquement décrit dans le code C++

PEIDA++: Contexte de développement et difficultés

- Évolution rapide de la spécification du C++
- Implémentation partielle/inexistante pour de nombreuses spécifications du C++, les exceptions par exemple
- Environnement hétérogène (AIX, HP au CC-IN2P3 , OSF au chili, Linux ...)
- Développement de composantes pour pallier à l'absence d'outils appropriés (AutoDoc par exemple)
- Difficultés d'adaptation pour une partie des physiciens

PEIDA++: Mise en oeuvre réussie de la conception objet dans un contexte de production

- Plusieurs tera-octets de données image et de courbes de lumière ont été traités avec succès
- Performances (temps CPU, mémoire ...) maintenues, comparées aux programmes antérieures (C/Fortran), alors que de nombreuses fonctionalités ont été ajoutées
- Acquisition d'une expertise dans le domaine de l'application du C++ au calcul lourd, dans un contexte de production intensive

SOPHYA

- Une nouvelle bibliothèque de classes C++, développé à partir de 1999, en utilisant l'expérience acquise sur EROS/PEIDA++
- Recherche de la simplicité (relative) d'utilisation , tout en maintenant des performances comparables avec le C/Fortran
- Développée initialement comme composante de base des logiciels Planck
- Contributions de
  - *E. Aubourg, S. Henrot-Versillé, A. Kim, G. Lemeur, C. Magneville, B. Revenu, F. Touze ...*

http://www.sophya.org

- Évaluation de quelques bibliothèques de classes (1998-1999)
  - math.h++ et lapack.h++ de Rogue Wave http://www.roguewave.com/
  - Blitz++ http://www.oonumerics.org/blitz/, TNT http://math.nist.gov/tnt/
- Quelques projets actifs actuellement (2005)
  - ATLAS http://math-atlas.sourceforge.net/
  - C++/BOOST http://www.boost.org/
  - IML++ (Iterative Methods Library) http://math.nist.gov/iml++/
  - CLHEP http://wwwasd.web.cern.ch/wwwasd/lhc++/clhep/
  - Blitz++ (?) , POOMA (?) http://acts.nersc.gov/pooma/

SOPHYA : vue générale

- Les conteneurs capables de représenter et gérer différentes structures de données (matrices et vecteurs par exemple)
- les entrées-sortie : persistance natif SOPHYA (PPF), formats d'échanges (texte/ascii, FITS, et peut-être HDF ultérieurement
- les algorithmes : Utilisation et encapsulation de librairies (C/Fortran) dans la mesure du possible (FFT, Lapack, ...)

Gestion mémoire et exceptions

- Création d'objets sur la pile : La destruction est automatique et compatible avec les exceptions (throw)
  - L'allocation dynamique doit être effectué dans les constructeurs
  - Gestion automatique de la mémoire, lors de l'utilisation des classes
- Beaucoup plus efficace qu'une allocation dynamique pour des objets de petite taille.
  - Gain de temps d'un facteur ≈10 lors de la création d'objets de petite taille dans des boucles

Objets de grande taille: partage de référence

- Pour éviter des recopies gourmandes en temps, il faut mettre en place un mécanisme de partage de référence pour les objets contenant de gros volumes de données
- Régle adoptée dans SOPHYA (en général) :
  - Le constructeur de copie partage les données
  - L'opérateur égal (=) effectue une copie (duplique les données)

Classes de bases pour la gestion du partage de référence dans SOPHYA

- NDataBlock<T> : Gestion d'un bloc de données numériques en mémoire, avec comptage/partage de référence
- SegDataBlock<T> : Gestion de structures segmentées de données en mémoire pour tout type T, avec comptage/partage de référence
- SwSegDataBlock<T> : Gestion de structures segmentées de données en sur une zone de swap pour tout type T, avec comptage/partage de référence

SOPHYA : modules (packages)

- BaseTools : Définition de l'architecture et des classes de base
- TArray : Classes template de tableaux (jusqu'à 5 D), Matrices, Vecteurs pour tous les types numériques, y compris complex<T>
- HiStats : Classes d'histogrammes et de tables (DataTable/NTuple)
- SkyMap : Représentation pixelisés de carte sphériques
- NTools : algorithmes numériques de base (Ajustements, FFT ...)
- Samba : géométrie 3D, transformé en Ylm, ...
- SysTools, SUtils : Utilitaires et interfaces avec les services du système (threads, chargement dynamique, interpreteur ...)

SOPHYA : librairies externes et packages interfaces

- FitsIOServer : Gestion des fichiers FITS à travers c-fitsio
- IFFTW : Transformé de Fourier, à travers la librairie C FFTW, conforme à l'interface FFTServerInterface définie dans NTools
- LinAlg : Interface partielle avec Lapack
- XAstroPack : Calculs temps/coordonnées astronomiques, à travers la librairie de XEphem
- MinuitAdapt : Interface avec les routines de minimisation de Minuit

- SophyaLib:
  - 300 fichiers (.cc .h .c) , >80 kl, 2,3 MO
  - >150 classes, dont de nombreuses classes template
- SophyaExtLib
  - 40 fichiers (.cc .h .c) , > 13 kl, 400 kO
  - 20 classes
- PI/piapp
  - 280 fichiers (.cc .h .c) , 70 kl, 2 MO
  - >130 classes

Mécanismes de Persistance objet dans SOPHYA (PPF)

- Séparation claire entre les objets (données, opérations) et la persistance
  - Agents/Objets délégués assurant la transcription (hérite de PPersist)
- Aucune (presque) contrainte sur les objets
  - Gère par exemple la persistance de std::vector<T>
- Écriture séquentielle, lecture séquentielle + fonctionalités d'accès direct (tag de positionnement et tag d'identification nommé)
  - Possibilité d'utilisation de flots à travers le réseau
- Gestion des objets avec partage de référence

Structure des fichiers PPF

- Ensemble de données des type de base (int/unsigned x1 x2 x4 x8, float x4 x8, complex x4 x8, vecteurs (tableaux) de ces mêmes types et chaînes de caractères (string), tous identifiés (tag)
- Fichiers portable (format IEEE et gestion du byte-swap)
- Tag de structuration :
  - `<PPS_OBJECT>, <PPS_ENDOBJECT> , <PPS_REFERENCE> , <PPS_NAMETAG_MARK>, <PPS_NAMETAG_TABLE> ,<PPS_POSTAG_MARK> , <PPS_POSTAG_TABLE> ,<PPS_EOF> ...`
- Structure de base : Objet pouvant contenir des données de type simple, des tableaux, d'autres objets ou références d'objets

Exemple d'utilisation (1)

```
---------- Ecriture ---------
// We create a integer array SizeX=7, SizeY=5
TArray<int_4> ia(7,5);
// We fill it ....
// We extract a subarray ib
TArray<int_4> ib = ia(Range(0,3), Range(3,4),
Range(0));
TArray<r_4> fc(10,8), fd(5,5);
// Fill the arrays ....
cout << " >>>>>> Writing in arrt.ppf <<<<<<< " << endl;
POutPersist pos("arrt.ppf");
pos << ia << ib;
pos << fc;
pos << PPFNameTag("FD") << fd ;
```

Exemple d'utilisation (2)

```
---------- Lecture ---------
cout << " >>>>>>> Reading from arrt.ppf " << endl;
PInPersist pis("arrt.ppf");
TArray<int_4> iaa, ibb;
TArray<r_4> fcc(10,8), fdd(5,5);
// We read the three arrays from the stream
pis >> iaa >> ibb;
pis >> PPFNameTag("FD") >> fdd;
```

Plateformes et compilateurs

- Linux / g++ (3.x)
- MacOSX 10.3 (Darwin) / g++ (3.3)
- HP Tru64 (OSF1) / cxx (6.3)
- SGI IRIX64 / CC (7.x)
- Linux / icc (8.0) compilateur Intel

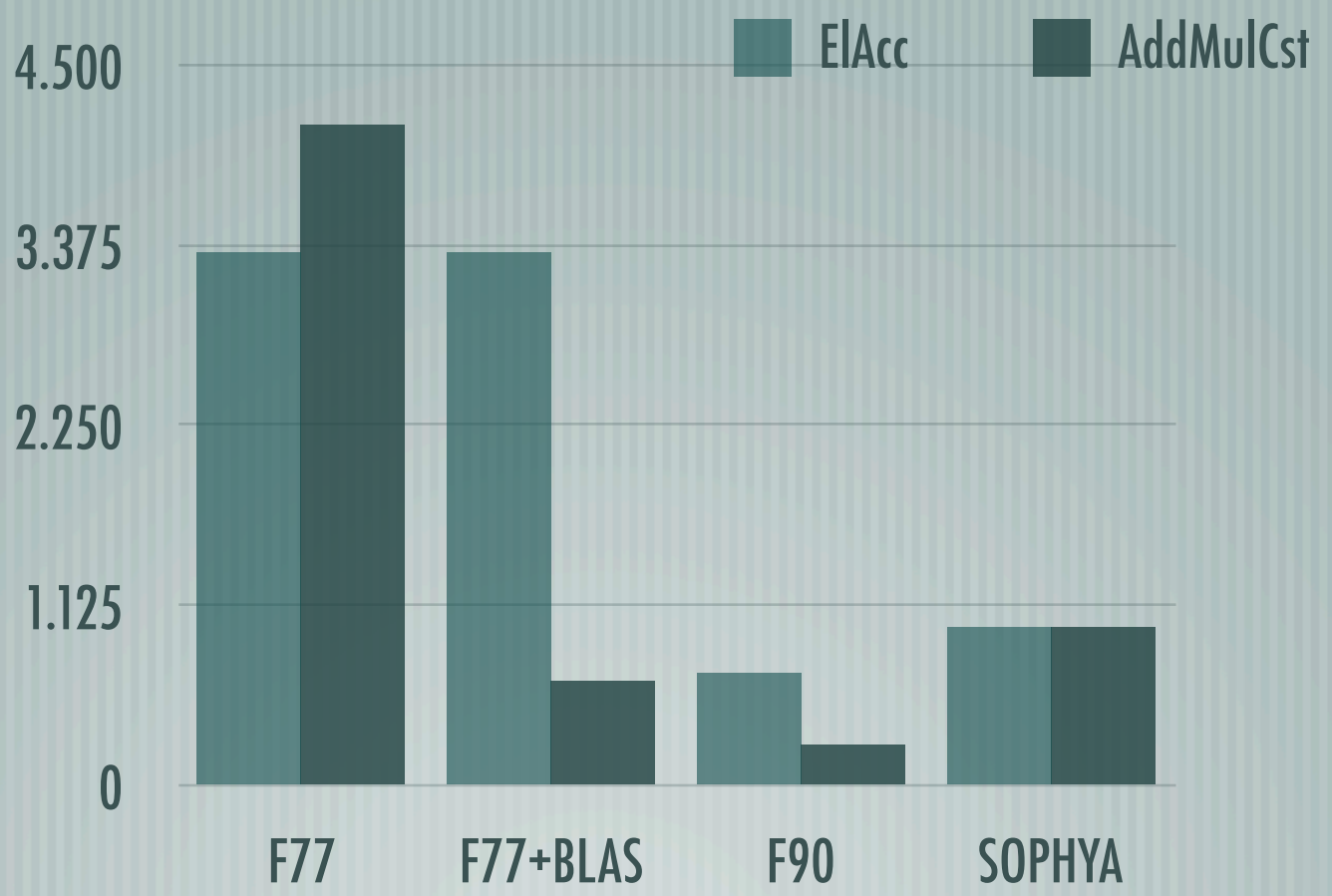
Performances : plateformes de référence

- eros3: Intel(R) Xeon(TM) CPU @ 2.40GHz
  - Linux , g++ 3.3 , BLAS compilé avec g77
  - cpupower (-O3) : 1→880 , 2→1160 , 2→1800 MFLOPS
- asc: ES47 Chip Alpha EV7 @ 1 GHz
  - OSF1, cxx 6.5 , BLAS optimisé
  - cpupower (-O3) : 1→500 , 2→840 , 2→1270 MFLOP

Performances - T1

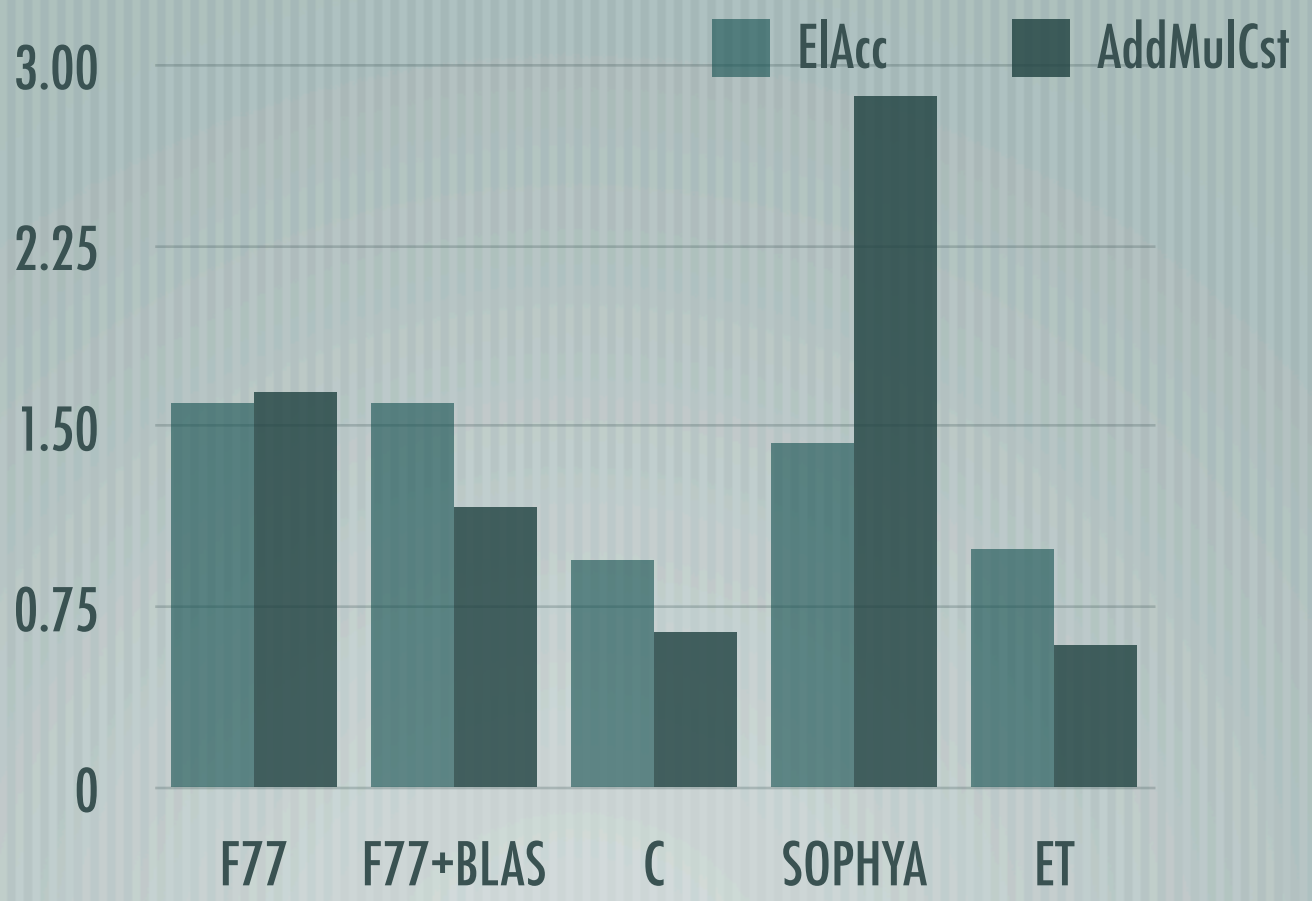
- Tests: Matrices (double) 1000x500, boucle de 50 fois
  - ElAcc : Accès aux éléments, initialisation
  - AddMulCst: Calcul d'une expression
    - mx = mx1*c1 + mx2*c2 + mx3*c3
- Comparaison SOPHYA (TMatrix<r_8>), F77 , F77 + BLAS, F90, C, ET (Expression template)

T1 sur asc (TCPU en secondes)



Comparaison SOPHYA::TMatrix<r_8> / Fortran

T1 sur eros3 (TCPU en secondes)

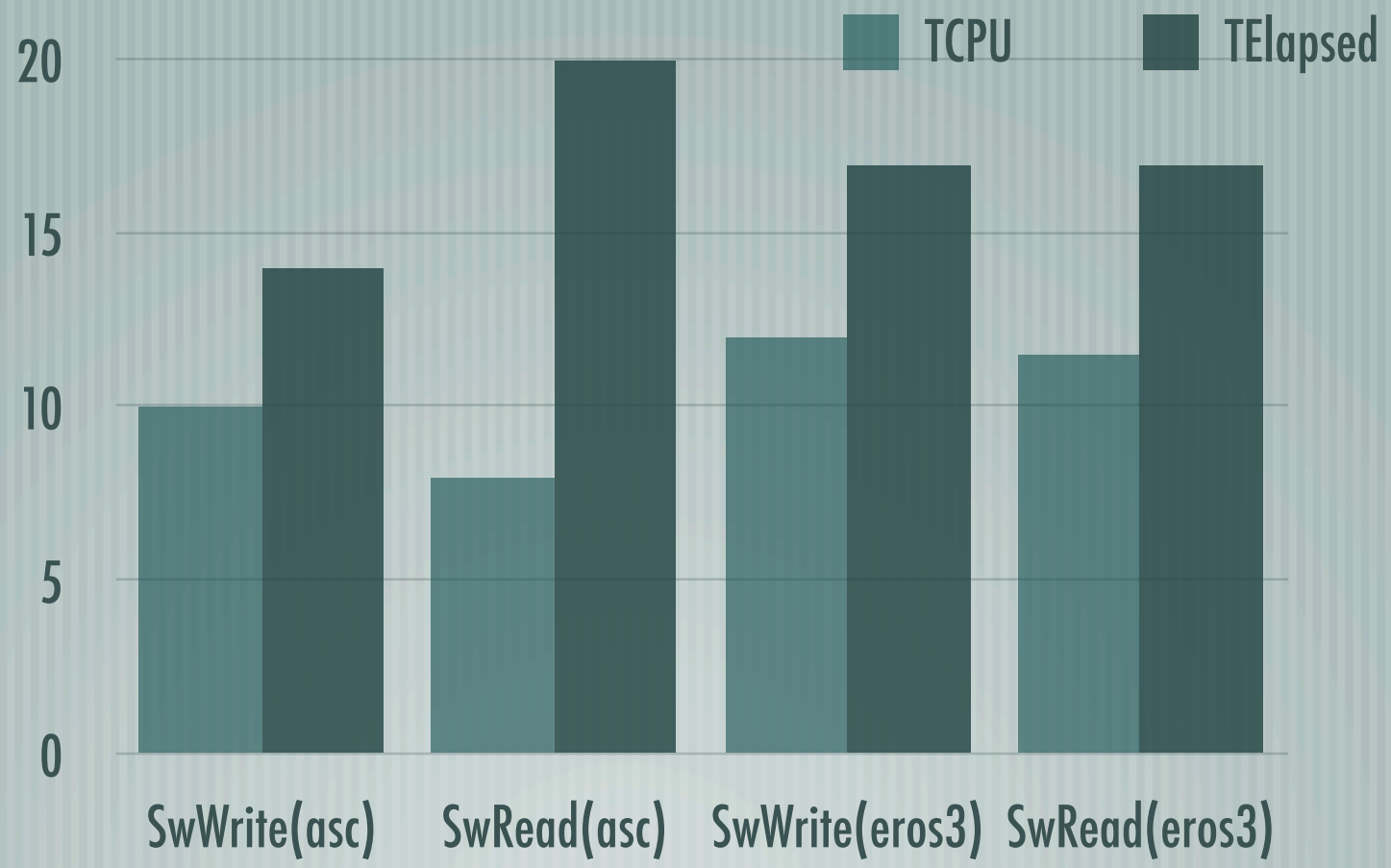


Comparaison SOPHYA::TMatrix<r_8> / ET/C /Fortran

Performances - T2

- Tests: Création/écriture et parcours/lecture d'une table de 10000000 (10^7) lignes et 7 colonnes (int+6xdouble)
  - SwWrite : Création et écriture de la table
  - SwRead : Parcours et lecture de la table, calcul moyennes et écart-type
- Utilisation de la classe SOPHYA::SwapDataTable (gestion des données en mode swap sur stream PPF)
- Taille total du fichier : 520 MO

T2 sur asc et eros3 (disques locaux)

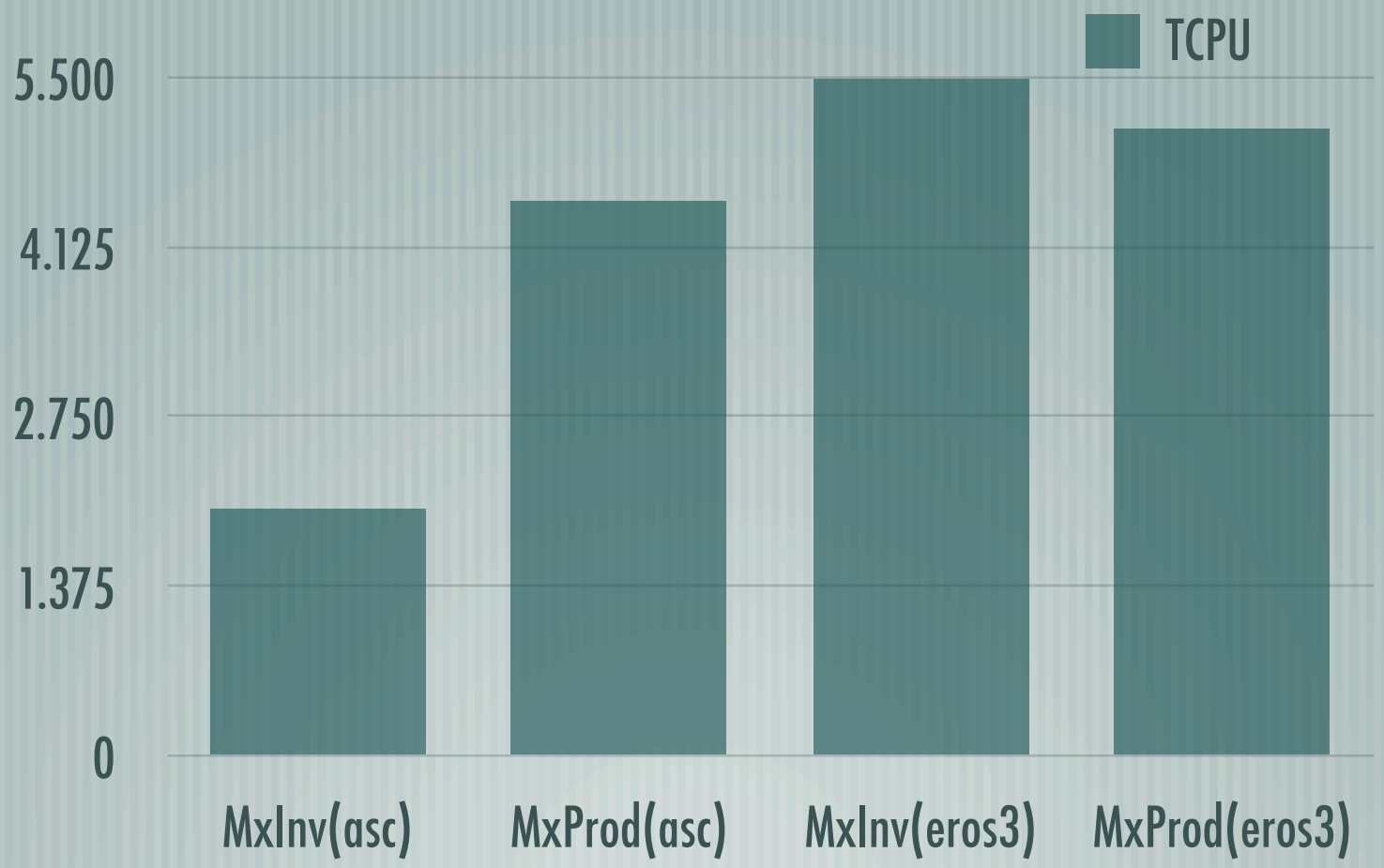


Réarrangement/conversion de données et entrée-sortie

Performances - T3

- Tests: Inversion et multiplication de matrices
  - MxInv : Inversion d'une matrice (double) 1000x1000
  - MxProd : Multiplication de deux matrices 1000x1000 (2 GFLOP)
- Performance de l'inversion à travers la classe SOPHYA::LapackServer<T>
- Multiplication matricielle SOPHYA::TMatrix<T>, mx3 = mx1*mx2;

T3 sur asc et eros3



Inversion et multiplication de matrices

piapp (1)

- Programme d'analyse interactive graphique, basé sur SOPHYA et la librairie graphique PI
- Définition d'une architecture d'application autour d'une classe de gestion d'objets nommés (NamedObjMgr) et d'une classe fournissant les services graphiques (PIStdImgApp)
- Des classes adaptateurs, héritant de NObjMgrAdapter prennent en charge les objets gérés NamedObjMgr.

piapp (2)

- Bonne interactivité graphique
- De nombreuses représentations graphiques 2D
- Visualisation d'image
- Quelques représentations 3D (nuage de points, surface, ...)
- La presque totalité des représentations graphiques sont réalisées à travers deux ou trois interfaces (abstraites)

piapp (3)

- Un interpréteur de commande inspiré de c-shell (peut prendre en charge d'autres interpréteurs, Tcl par exemple)
- Peut effectuer la compilation au vol (C/C++) et gère le chargement dynamique de fonctions et de modules
- Calcul et représentation d'expressions (à la PAW), étendu à tous les objets (utilise la compilation au vol)
- Permet l'execution de bout de code C++, opérant sur les objets gérés par l'application
- Application multi-thread , permet l'execution de commandes sur des threads séparés