

GPUification avec OpenAcc

section efficace de capture d'électrons dans les supernovæ

Vincent LAFAGE
Frédéric BASTIER

¹D2I, Institut de Physique Nucléaire
Université Paris-Sud



3 octobre 2018

```
! Electron capture rates based on finite temperature
! Hartree-Fock and finite temperature random-phase
! approximation using Skyrme interactions
!
! Electron capture code based on mixed version between
! Fortran (ftchrpa.f) and C codes (eccalc.cpp, mult_me.cpp,
! multred.cpp, etc...)
!
! 21/11/2008 Nils Paar
```

Stellar electron-capture rates calculated with the finite-temperature relativistic random-phase approximation

Y. F. Niu¹, N. Paar², D. Vretenar², and J. Meng^{3,1,4*}

¹State Key Laboratory of Nuclear Physics and Technology,

Calculation of stellar electron-capture cross sections on nuclei based on microscopic Skyrme functionals

N. Paar

Physics Department, Faculty of Science, University of Zagreb, Croatia

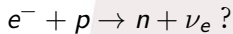
G. Colò

Dipartimento di Fisica dell'Università degli Studi and INFN

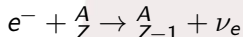
[nucl-th] 12 Apr 2011

We int
electron
The find

IPNO の理論部
計算天体物理学
こうせい こうせい
恒星内元素合成：
中性子星になるとき



⇒



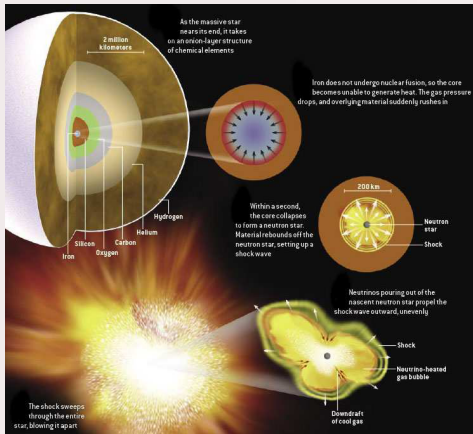
千原子核の種類

温度：[0.5, 5] MeV,

50 ステップ

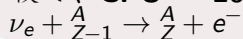
⇒ CPU 千年！

逆原子核反応も大切：



⇒ スピードアップ° > 55

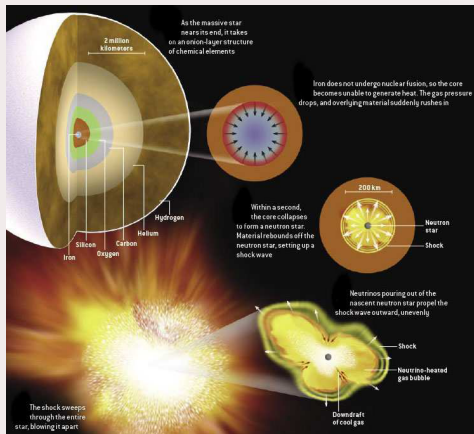
⇒ 後で、CPU 20 年だけ



Capture d'électrons & SuperNovæ

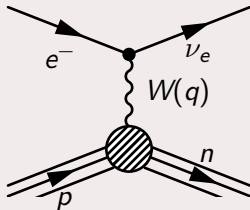
Groupe théorie IPNO
Simulation :
1000 types de noyaux
50 valeurs de T
⇒ 1000 ans de CPU

- * *profiling*
- * audit
- bibliothèques
- * gestion mémoire
- * vectorisation



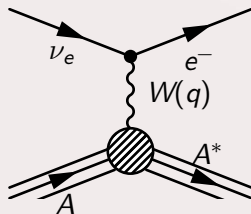
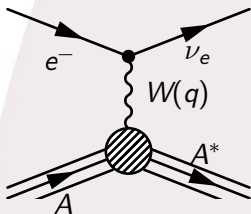
⇒ facteur > 55

⇒ 20 ans de CPU après accélération



« Quand la pression gravitationnelle dépasse la pression de dégénérescence des électrons, ceux-ci sont capturés par les protons »

Processus effectif



$$\langle p_e, n_{pA} | \mathcal{M} | \cos \theta_\nu, n_{nA^*}, r, J^\pi; T \rangle > 0$$

$$\langle p_e, n_{pA}(J^\pi; T) | \mathcal{M} | \cos \theta_\nu, n_{nA^*}(J^\pi; T), r \rangle$$

$$\forall J^\pi, T \quad \sigma \propto \int dp_e d \cos \theta_\nu dr \sum_{n_{pA}, n_{nA^*}} |\langle p_e, n_{pA} | \mathcal{M} | \cos \theta_\nu, n_{nA^*}, r \rangle|^2$$

(en incluant les fonctions de structures)

Structure du programme

```
(... boucles sur J, Π ...)           ! 6
(... calcul de structure nucléaire selon un modèle RPA : ...)
(... occupations des niveaux d'énergies, fonctions d'ondes wf / dwf ...)
do idxenergy0 = 0, nbenergystep - 1  ! 150 énergies electron
  do energyc = 1, nconf               ! 153 / 405 / 540 états d'énergie du noyau
    do k_simps = 0, n_simps           ! 31 angles electron-neutrino
      do pairc = 1, nconf             ! 153 / 405 / 540 états d'énergie du noyau T < 1 MeV
                                     ! 182 / 476 / 637 états d'énergie du noyau T = 2 MeV
                                     ! 232 / 609 / 821 états d'énergie du noyau T = 3 MeV
                                     ! 386 / 1041 / 1448 états d'énergie du noyau T = 4 MeV
      do i = 1, maxr                  ! 168 mailles radiales de fonction d'onde
        ... à fond de boucles, fonctions de bessel sphériques : > 92% du temps
```

$$j_e(qr)$$

Bessel Sphérique

$$\begin{aligned}
 j_0(z) &= \frac{\sin z}{z} \\
 j_1(z) &= \frac{\sin z}{z^2} - \frac{\cos z}{z} \\
 j_2(z) &= \left(\frac{3}{z^3} - \frac{1}{z} \right) \sin z - \frac{3}{z^2} \cos z \\
 &\dots
 \end{aligned}$$

Plus une fonction a de zéros, plus elle est difficile à évaluer précisément
Là, une infinité de zéros...

Benchmark de Bessel Sphérique

langage	algo	valeur	correct ?	-03 (ns)
Mathematica		0.06596800707652196074158...		
C++	g++ tr1::sph_bessel_L	0.0659680070765219607 530	62	794
C++	boost::sph_bessel	0.06596800707652196 45	53	
C	gsl_sf_bessel_jl = ROOT	0.06596800707652196	56	1065
C	NR sphbes $\mu=10^{-1}$ rtpio+O3	0.06596800707652196	56	559
C++	g++ tr1::sph_bessel	0.0659680070765219 367	51	661
Fortran	SPHBR	0.06596800707652196 4	54	3240
Fortran	SPHBES	0.0659680070765219 51	52	747
Fortran	SPHBES $\mu=10^{-10}$ rtpio+	0.06596800707652 67385 ???	52	669
Fortran	SBESJY	0.06596800707652 2006	50	439
Fortran	SBESJH	0.06596800707652196 4	54	3130
Fortran	SBESJH	0.06596800707652 2020	50	
Fortran	SBESJ	0.0659680070765219 51	52	440
C	NR sphbes $\mu=10^{-10}$ rtpio+	0.06596800707652 67385	43	699
C	NR sphbes $\mu=10^{-10}$	0.06596800 511243080	25	699

DSE Bessel sphérique

CUDA

$$j_n(z) = z^n \sum_{k=0}^{\infty} \frac{(\frac{1}{2}z^2)^k}{k!(2n+2k+1)!!}$$

```

module bessel
  use, intrinsic :: ISO_C_BINDING
  implicit none
  integer, parameter :: pr = c_double ! precision
  integer, parameter :: &
    nmax = 19, &
    lmax = 10
  integer(kind=c_int), protected, bind (C, NAME = "lsize") :: C_lsize = lmax
  integer(kind=c_int), protected, bind (C, NAME = "nsize") :: C_nsize = nmax

  integer, parameter :: ep = 16
  real (ep), parameter :: &
    Qpi = 4 * atan (1.0_ep), &
    Qlog2 = log (2.0_ep)

  real (c_double), dimension (0:lmax, 0:nmax), save, protected, target, bind (C, NAME = "coeff")
  finecoeff = reshape (source = real (exp (- (/ &
    ((log_gamma (n+1.0_ep) + &
      log_gamma (n+1+1.5_ep) + (2*n+1+1) * Qlog2 - log (Qpi) / 2.0_ep, l = 0, lmax), n = 0
    shape = shape (finecoeff))
  type (C_PTR), save, protected, bind (C, NAME = "coeffBessel0") :: coeffBesselPtr != C_LOC
  ...

```

Constantes

Constante	Initiale	Actuelle		Δ	bits	incertitude	bits
TanCab	0.25	0.23125(87)		7.50e-02	3.7	3.76e-03	8.1
neutronmass	939.0	939.565 4133(58)	MeV	6.02e-04	10.7	2.24e-08	27.3
alphainv	137.0	137.035 999 139(31)		2.63e-04	11.9	3.21e-10	32.0
hbc	197.3	197.3269788(12)	MeV·fm	1.37e-04	12.8	2.23e-08	27.3
Vud	0.97419	0.97427(21)		8.21e-05	13.6	2.16e-04	12.2
dblMasse...	0.04823	0.0482264271(32)		7.41e-05	13.7	6.70e-08	24.8
Gf	1.16639e-11	1.166 3787e-11(6)	MeV ⁻²	2.23e-05	15.5	4.29e-05	20.9
protonmass	938.2796	938.272 0813(58)	MeV	8.05e-06	16.9	2.24e-08	27.3
emass	0.511003	0.5109989461(31)	MeV	7.97e-06	16.9	2.15e-08	27.3
clight	2.99792e23	2.99792458e23	fm·s ⁻¹	1.53e-06	19.3		

Constantes 2019

Constante	Initiale	Actuelle		Δ	bits	incertitude	bits
TanCab	0.25	0.23125(87)		7.50e-02	3.7	3.76e-03	8.1
neutronmass	939.0	939.565 4133(58)	MeV	6.02e-04	10.7	2.24e-08	27.3
alphainv	137.0	137.035 999 139(31)		2.63e-04	11.9	3.21e-10	32.0
hbc	197.3	197.3269799666552...	MeV·fm	1.37e-04	12.8		
Vud	0.97419	0.97427(21)		8.21e-05	13.6	2.16e-04	12.2
dbIMasse...	0.04823	0.0482264271(32)		7.41e-05	13.7	6.70e-08	24.8
Gf	1.16639e-11	1.166 3787e-11(6)	MeV ⁻²	2.23e-05	15.5	4.29e-05	20.9
protonmass	938.2796	938.272 0813(58)	MeV	8.05e-06	16.9	2.24e-08	27.3
emass	0.511003	0.5109989461(31)	MeV	7.97e-06	16.9	2.15e-08	27.3
clight	2.99792e23	2.99792458e23	fm·s ⁻¹	1.53e-06	19.3		
<i>h</i>		6.62607015e-34	J·s				
<i>e</i>		1.602176638e-19	C				
<i>N_A</i>		6.02214076e23	mol				

Projet de résolution « Sur la révision du Système international d'unités (SI) »
<https://www.bipm.org/utils/fr/pdf/CGPM/Draft-Resolution-A-FR.pdf>
 3-16 novembre 2018 \implies 20 mai 2019

Projet de résolution A – 26^e réunion de la CGPM (13/10/2018)

Projet de résolution A

Sur la révision du Système international d'unités (SI)

La Conférence générale des poids et mesures (CGPM), à sa 26^e réunion, **considérant**

- qu'il est essentiel de disposer d'un Système international d'unités accessible dans le monde entier, pour le commerce international, l'

```

module physconst
  use mod_precision
  use, intrinsic :: ISO_C_BINDING
  implicit none
  real (pr), parameter :: &
    pi          = 4 * atan (1.0_pr), & !  $\pi$ , Archimedes' constant
    sqrtpi     = sqrt (pi),          & !  $\sqrt{\pi}$ 
    sq4pi      = sqrt (4 * pi),      & !  $\sqrt{4\pi}$ 
    Gf         = 1.1663787e-11_pr,    & ! Gf, Fermi constant ! 1.166 3787(6)  $\times 10^{-11}$  MeV-2 (2014 CODATA)
    hbc        = 197.3269788_pr,      & !  $\hbar c$  ! 197.326 9788(12) MeV·fm (2014 CODATA)
    alphasinv  = 137.035999139_pr,    & !  $1/\alpha_{em}$  ! 137.035 999 139(31) (2014 CODATA)
    protonmass = 938.2720813_pr,      & !  $m_p$  ! 938.272 0813(58) MeV (2014 CODATA)
    neutronmass = 939.5654133_pr,     & !  $m_n$  ! 939.565 4133(58) MeV (2014 CODATA)
    TanCab     = 0.23129_pr,          & !  $\tan \theta_{Cab}$ , Cabibbo angle ! Is it  $\tan \theta_{Cab}$ ? => 0.22534/0.9742
    thetaC     = atan (TanCab),       & !  $\theta_{Cab}$ , Cabibbo angle
    emass      = 0.5109989461_pr,     & !  $m_e$  ! 0.510 998 9461(31) MeV (2014 CODATA)
    clight     = 2.99792458e23_pr,    & ! c speed of light in vacuum ! 2.99792458  $\times 10^{23}$  fm·s-1 (1983 CGPM)
    Vud        = 0.97427_pr,          & !  $V_{ud}$ , Cabibbo-Kobayashi-Maskawa up-down coupling matrix element
    gv         = 1.0_pr,              & !  $g_v$ 
    Na         = 6.022140857e23_pr,    & ! Na, Avogadro constant ! 6.022 140 857(74)  $\times 10^{23}$  mol-1 (2014 CODATA recon
    deltanp    = 1.29333205_pr,        & !  $A_{np} = m_n - m_p$  ! 1.293 332 05(48) MeV (2014 CODATA)
    Rydberg    = emass / alphasinv*2 / 2.0_pr, & !  $R_{hc} = 13.605 693 009(84) \times 10^{-6}$  MeV (2014 CODATA)
    deltanH    = 0.78227_pr,          & !  $A_{n^1H} = m_n - m_p - m_e$  ! deltanp - emass = 0.78233310(48) or 0.78234670(4
ion energy (2014 CODATA)
    ampi       = 138.0_pr,            & ! around pion masses = 139.57018; //  $\pm 0.00035$  GeV //  $\pi^{++}$  (charged

```

```

module physconst
  use, intrinsic :: ISO_C_BINDING
  implicit none
  real (kind=C_DOUBLE), parameter :: & ! (2014 CODATA)
    pi          = 4 * atan (1.d0), & ! , Archimedes' constant
    sqrtpi     = sqrt (pi), & ! √
    sq4pi      = sqrt (4 * pi), & ! √(4 )
    Gf         = 1.1663787d-11, & ! GF, Fermi constant ! 1.166 3787(6) × 10-11 MeV2
    hbc        = 197.3269788d0, & ! ! 197.326 9788(12) MeV=fm
    alphainv   = 137.035999139d0, & ! 1/ ! 137.035 999 139(31)
    protonmass = 938.2720813d0, & ! m ! 938.272 0813(58) MeV
    neutronmass = 939.5654133d0, & ! m ! 939.565 4133(58) MeV
    TanCab     = 0.23129d0, & ! tan Cab, Cabibbo angle => 0.22534/0.97427
    thetaC     = atan (TanCab), & ! Cab, Cabibbo angle
    emass      = 0.5109989461d0, & ! m ! 0.510 998 9461(31) MeV
    clight     = 2.99792458d23, & ! speed of light in vacuum ! 2.99792458 × 1023 fm*s
    Vud        = 0.97427d0, & ! V , Cabibbo-Kobayashi-Maskawa up-down coupling matr
    gv         = 1.0d0, & ! g
    Na         = 6.022140857d23, & ! N , Avogadro constant ! 6.022 140 857(74) × 1023
    ...
    factco     = (Gf * cos (thetaC) * hbc)**2 * 1d16 ! factor for cross sections
    ...
  ! Error: PARAMETER attribute conflicts with BIND(C) attribute
  real(kind=C_DOUBLE), protected, bind (C, NAME = "hqc") :: C_hqc = hbc ! 197.32
  ...

```

Success story

- Code de capture d'électrons
9 kSLOC de Fortran, 5 kSLOC de C(++)
- Accélération avant $//^n$
 - ... facteur 12 (stockage plutôt que recalcul)
 - ... facteur **25** \Rightarrow **55** au total
 - utiliser les bibliothèques standard (nearbyint)
 - Spherical Bessel Benchmark (15 codes, 9 algos)
 - vectorisation des boucles...
- amélioration de la précision

Ces résultats sont le fruit d'une méthode :

- ⇒ on ne rentre pas dans 15 000 lignes de code comme dans un moulin : mise du code sous contrôle de version `svn`
- analyse statique `ftncheck`, `cppchecker`
 - * branches mortes (procédures, variables)
 - * métriques `sloccount`
 - analyse dynamique "profiling" `gprof`
 - * identification des goulets d'étranglements
 - * optimisation de fond de boucles
 - chasse aux problèmes de mémoire `valgrind`
 - typographie, indentation, documentation `doxygen`
 - pêche aux mauvaises pratiques numériques

D. GOLDBERG, *What every computer scientist should know about floating point arithmetic*

- * constantification des constantes : extraction des constantes en dur, uniformisation : combien de valeurs de π distinctes ?
- * accélération (Horner, Richardson, stockage intermédiaire...)

Parallélisation ?

La parallélisation a l'air facile \Rightarrow OpenMP

```
(... boucles sur J,  $\Pi$  ...)           ! 6
!$OMP PARALLEL PRIVATE (Eelectron, ecsum) SHARED (Energy_electron, sigma)
!$ & COPYIN (/energyidx/, /rpamix/, /bwf/, /occupa/, /qval/, /bdiam/,
!$ & /bqwf/, /bwu2/, /bnri/, /bnrir/, /bwu1/, /bwuir/, /blecp1/,
!$ & /blecp2/, /bpt1/)
do idxenergy0 = 0, nbenergystep - 1    ! 150 énergies electron
  do energyc = 1, nconf                 ! 153 / 405 / 540 états d'énergie du noyau
    do k_simps = 0, n_simps             ! 31 angles electron-neutrino
      do pairc = 1, nconf               ! 153 / 405 / 540 états d'énergie du noyau
        do i = 1, maxr                 ! 168 mailles radiales de fonction d'onde
          ... à fond de boucles, fonctions de bessel sphériques : > 92% du temps
```

\Rightarrow segmentation fault :(

IDRIS \Rightarrow « Convertissez tout dans un seul langage »
Fortran 90, validation des résultats, ajout de OpenMP

\Rightarrow segmentation fault :(

- * Faut-il simplifier le code ?
- * Dur avec des threads, facile avec des process
- * Threads pas si équivalents en durée...
- * Transformer le code pour exprimer une transformée rapide de Bessel Sphérique à la FFT ?
- * intégration numérique reposant des produits de matrice \Rightarrow utilisons le GPU
CUDA, OpenCL, OpenAcc?


```

subroutine radpoint_array (j0)
  implicit none
  integer, intent (in) :: j0
  integer :: k_simps, pairc, idxenergy, energyc, maj ! dummy indices
  integer :: Jmin, Jmax
  Jmin = max (j0-1, 0)
  Jmax = j0+1

  allocate (Rad_point1_array (1:nconf, 0:n_simps, 1:nconf, 0:nbenergystep-1, Jmin:Jmax))

  build10: do maj = Jmin, Jmax
    build11: do idxEnergy = 0, nbenergystep-1
      build12: do energyc = 1, nconf
        if (mask_kinematics (energyc, idxEnergy)) then
          build13: do k_simps = 0, n_simps
            build14: do pairc = 1, nconf
              Rad_point1_array (pairc, k_simps, energyc, idxEnergy, maj) = &
                sum (wfldwf2_array (1:maxr, pairc) * Bess_f_array (1:maxr, k_simps, energyc,
              end do build14
            end do build13
          end if
        end do build12
      end do build11
    end do build10

    Rad_point1_array (:, :, :, :, :) = Rad_point1_array (:, :, :, :, :) - Rad_point2A_array
  end subroutine radpoint_array

```

```

subroutine radpoint_array (j0)
  implicit none
  integer, intent (in) :: j0
  integer :: k_simps, pairc, idxenergy, energyc, maj ! dummy indices
  integer :: Jmin, Jmax
  Jmin = max (j0-1, 0)
  Jmax = j0+1

  allocate (Rad_point1_array (1:nconf, 0:n_simps, 1:nconf, 0:nbenergystep-1, Jmin:Jmax))

  build10: do maj = Jmin, Jmax
    build11: do idxEnergy = 0, nbenergystep-1
      build12: do energyc = 1, nconf
        build13: do k_simps = 0, n_simps
          build14: do pairc = 1, nconf
            Rad_point1_array = reshape (matmul (transpose (wfidwf2_array (1:maxr, 1:nconf)),
              reshape (bess_f_array, (/maxr, (n_simps+1) * nconf * nbenergystep * (maxj+
                (/nconf, (n_simps+1), nconf, nbenergystep, (maxj+2))))))

            end do build14
          end do build13
        end do build12
      end do build11
    end do build10

    Rad_point1_array (:, :, :, :, :) = Rad_point1_array (:, :, :, :, :) - Rad_point2A_array
  end subroutine radpoint_array

```

```

subroutine radpoint_array (j0)
  implicit none
  integer, intent (in) :: j0
  integer :: k_simps, pairc, idxenergy, energyc, maj ! dummy indices
  integer :: Jmin, Jmax
  Jmin = max (j0-1, 0)
  Jmax = j0+1

  allocate (Rad_point1_array (1:nconf, 0:n_simps, 1:nconf, 0:nenergystep-1, Jmin:Jmax))

  build10: do maj = Jmin, Jmax
    build11: do idxEnergy = 0, nenergystep-1
      build12: do energyc = 1, nconf
        if (mask_kinematics (energyc, idxEnergy)) then
          build13: do k_simps = 0, n_simps
            build14: do pairc = 1, nconf
              Rad_point1_array (1:nconf, 0:n_simps, energyc, idxEnergy, maj) = &
                matmul (transpose (w1dwf2_array (1:maxr, 1:nconf)), &
                  Bess_f_array (1:maxr, 0:n_simps, energyc, idxEnergy, maj))
            end do build14
          end do build13
        end if
      end do build12
    end do build11
  end do build10

  Rad_point1_array (:, :, :, :, :) = Rad_point1_array (:, :, :, :, :) - Rad_point2A_array
end subroutine radpoint_array

```

```

subroutine radpoint_array (j0)
  implicit none
  integer, intent (in) :: j0
  integer :: k_simps, pairc, idxenergy, energyc, maj ! dummy indices
  integer :: Jmin, Jmax
  Jmin = max (j0-1, 0)
  Jmax = j0+1

  allocate (Rad_point1_array (1:nconf, 0:n_simps, 1:nconf, 0:nbenergystep-1, Jmin:Jmax))

  !$acc data present_or_create (mask_kinematics, Rad_point2B_array, Rad_point2A_array, Ra
  !$acc                               wf12_array, Bess_f_array, wf12inv_array, wf1dwf2_array)

  !$acc parallel loop collapse (3)
  build10: do maj = Jmin, Jmax
    build11: do idxEnergy = 0, nbenergystep-1
      build12: do energyc = 1, nconf
        if (mask_kinematics (energyc, idxEnergy)) then
  !$acc loop vector independent collapse (2)
          build13: do k_simps = 0, n_simps
            build14: do pairc = 1, nconf
              Rad_point1_array (pairc, k_simps, energyc, idxEnergy, maj) = &
                sum (wf1dwf2_array (1:maxr, pairc) * Bess_f_array (1:maxr, k_simps, energyc,
              end do build14
            end do build13
          end if
        end do build12
      end do build11
    end do build10
  !$acc end parallel loop

  !$acc parallel
  Rad_point1_array (: : : : :) = Rad_point1_array (: : : : :) - Rad_point2A_arr

```

	2013	r88	r153	r242						
	Intel	Intel	F90 Intel	new Intel	Intel ×	GNU	GNU ×	PGI	PGI ×	PGI openAcc
ftchrpa	39,31	39,31	39,31	39,31	39,14	83,19	83,15	69,35	69,27	69,41
eccalc	502,05	10,21	19,17	4,66	5,54	10,90	6,02	5,28	5,95	4,66
total	541	49,52	58,47	44,01	44,73	94,14	89,21	83,22	83,88	82,67
×eccalc		49,2		107,8	90,7		83,4		84,4	107,8
×eccalcr.				2,2	1,8		1,7		1,7	2,2

Speedup (news)

	2013	r88	r153	r242							
	Intel	Intel	F90 Intel	new Intel	Intel ×	GNU	GNU ×	PGI	PGI ×	PGI openAcc	PGI openAcc
ftchrpa	39,31 42,77	39,31	39,31	39,31	39,14	83,19	83,15	69,35	69,27	69,41	
eccalc total	502,05 541	10,21 49,52	19,17 58,47	4,66 44,01	5,54 44,73	10,90 94,14	6,02 89,21	5,28 83,22	5,95 83,88	4,66 82,67	5,04 53,83
×eccalc		49,2		107,8	90,7		83,4		84,4	107,8	
×eccalcr.				2,2	1,8		1,7		1,7	2,2	

- simple, double, étendue, quadruple
- généricité Fortran fondée sur `REAL(KIND=global_parameter)`
- différentes performances selon le compilateur
reste à faire du côté GNU (pas de vectorisation)
- pas testé sous OpenAcc

Tesla M2090 2011 \Rightarrow V100 $\times 7,7 ?$

- passage de ipngrid01 à llracp01
- passage de SLC 6 à CentOS 7
 \Rightarrow segmentation fault :(
- \Rightarrow recours à un container singularity
 \Rightarrow Merci Andrea! *bénéfice de la collab ASR / dev*
- overhead sur le premier appel au pilote NVidia : 4s!
- speed up sur la partie OpenAcc
 $\Rightarrow \times 1.7 \times 5 \Rightarrow \times 8.7 !!!$
- profiling (début) \Rightarrow expressions tableau \Rightarrow coût de l'initialisation
 \Rightarrow limiteS du compilateur

- différentes précision (simple, double, étendue, quadruple)
 - différentes performances selon le compilateur
 - pénalité d'abstraction de Fortran+C vers Fortran90
 - *Formula translator*
- + Tesla M2090 2011 \Rightarrow Tesla V100 $\times 8,7$
- + cc20 \Rightarrow cc70 profiling
- matmul !!!
- + cc > 35 : cuBLAS !!!
- + mémoire... (seulement 5 GB c'est limite \Rightarrow 16 GB)
gestion plus fine des régions mémoires
s'appuyer sur le masque cinématique pour diminuer
l'empreinte mémoires (mais perte de clarté)
- + nouvel OpenMPification