Ottimizzazione e parallelizzazione di codici di simulazione magnetoidrodinamica resistiva

Relatore : Ch.mo Prof. Salvatore Tucci
Correlatore : Ing. Salvatore Filippone
Tutor : Dott. Sergio Briguglio
Candidato : Michele Martone

Il codice MARST

Codice sviluppato da A. Bondeson, G. Vlad e H. Luetjens per **CRPP** (Centre de Recherches en Physique des Plasmas)-Lausanne e **ENEA**(Ente Nazionale Energia e Ambiente)-Frascati tra il 1989 e il 1992.

Impiegato per studi su *plasmi confinati magneticamente* in *tokamak* nell'ambito delle ricerche sulla *fusione termonucleare controllata*.

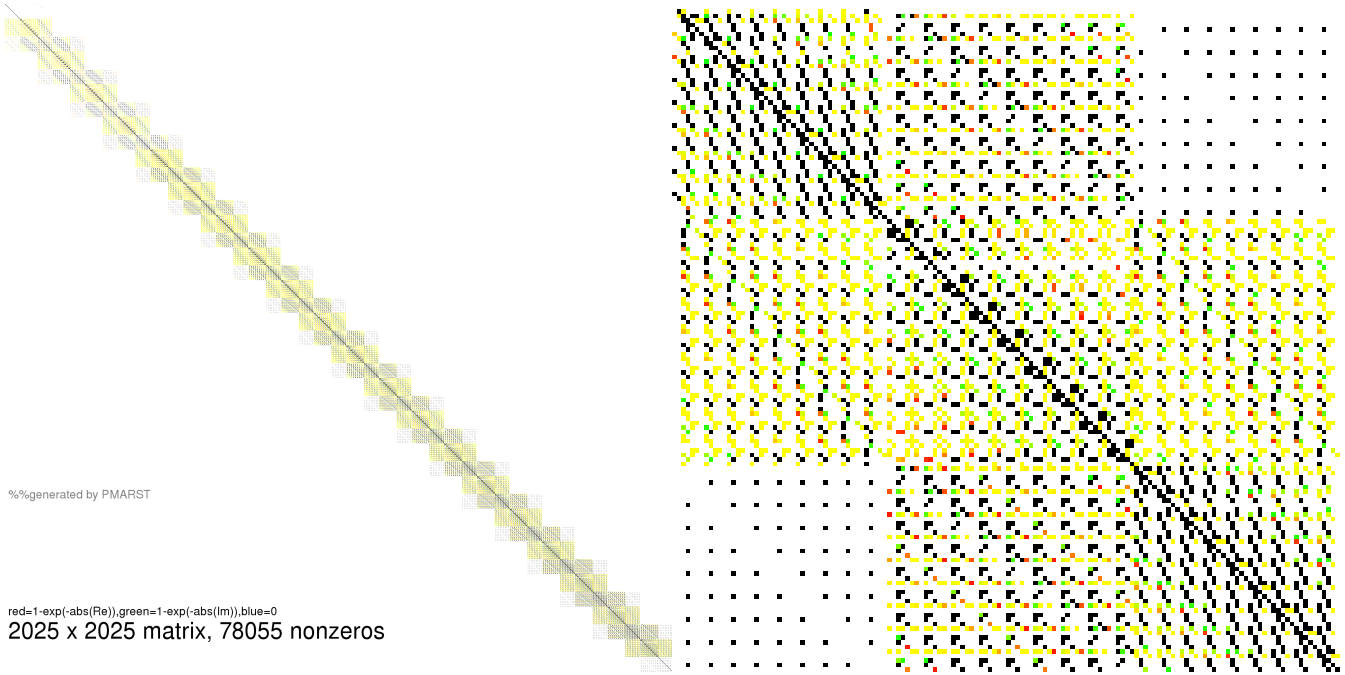
MAgnetohydrodynamic Resistive Spectrum with Timesteps

Risolve le equazioni della *magnetoidrodinamica resistiva* (*equazioni differenziali alle derivate parziali* linearizzate), che simulano l'*evoluzione temporale* del sistema. Una volta discretizzate, le equazioni si traducono in un sistema *algebrico lineare* [1] :

$$\left(A - \frac{1}{\Delta t} B\right) \mathbf{X}_t := -\frac{1}{\Delta t} B \mathbf{X}_{\mathbf{t-\Delta t}} \qquad (1)$$

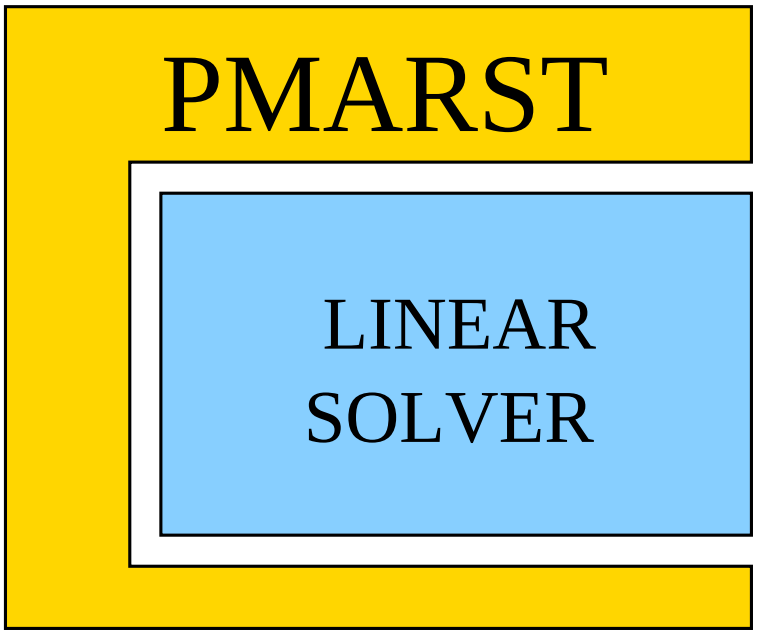
[1] Inoltre, si tratta di un problema *evolutivo*.

- matrici *tridiagonali bandate a blocchi densi*
- un caso *tipico* per applicazioni agli esperimenti *di prossima generazione*: rango $2 * 10^6$, con $1.6 * 10^{10}$ *nonzeri*
- problemi:
  - necessarie grandi risorse computazionali (tempo e, soprattutto, **memoria**)...
  - *risoluzione* non banale (**necessario** un metodo *ad hoc*)...



Obiettivi primari lavoro di Tirocinio/Tesi

- $\longrightarrow$ *parallelizzazione* del codice di **generazione** del sistema lineare
- $\longrightarrow$ *sostituibilità* del solutore del sistema lineare con uno *parallelo* e *scalabile* di **terze parti**

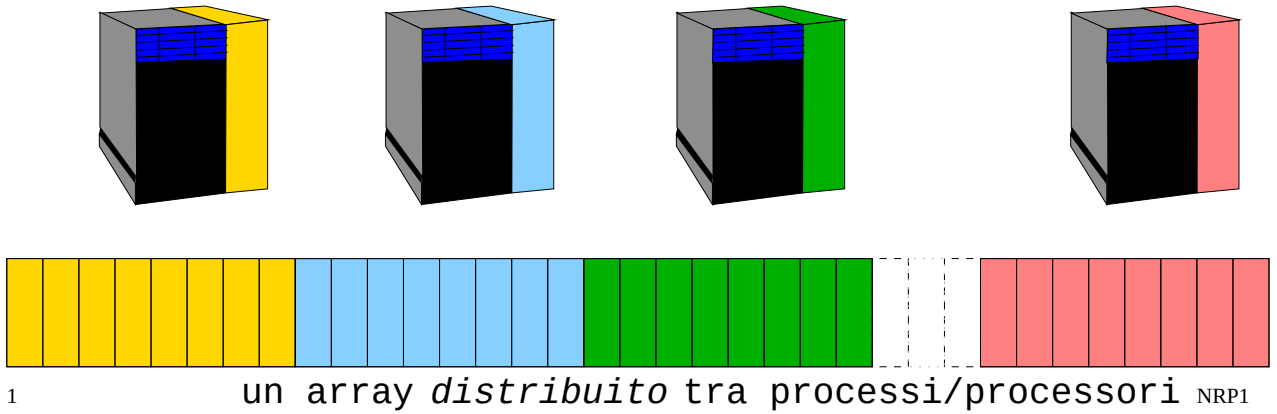


Obiettivi secondari lavoro di Tirocinio/Tesi

- Codice **portabile** :IBM AIX, GNU/Linux, ...
- Codice **manutenibile** dagli autori originali:
  - Per modifiche al *modello fisico* simulato
  - Per interfacciamento con codici *a particelle*

Un programma, più *nodi* di calcolo

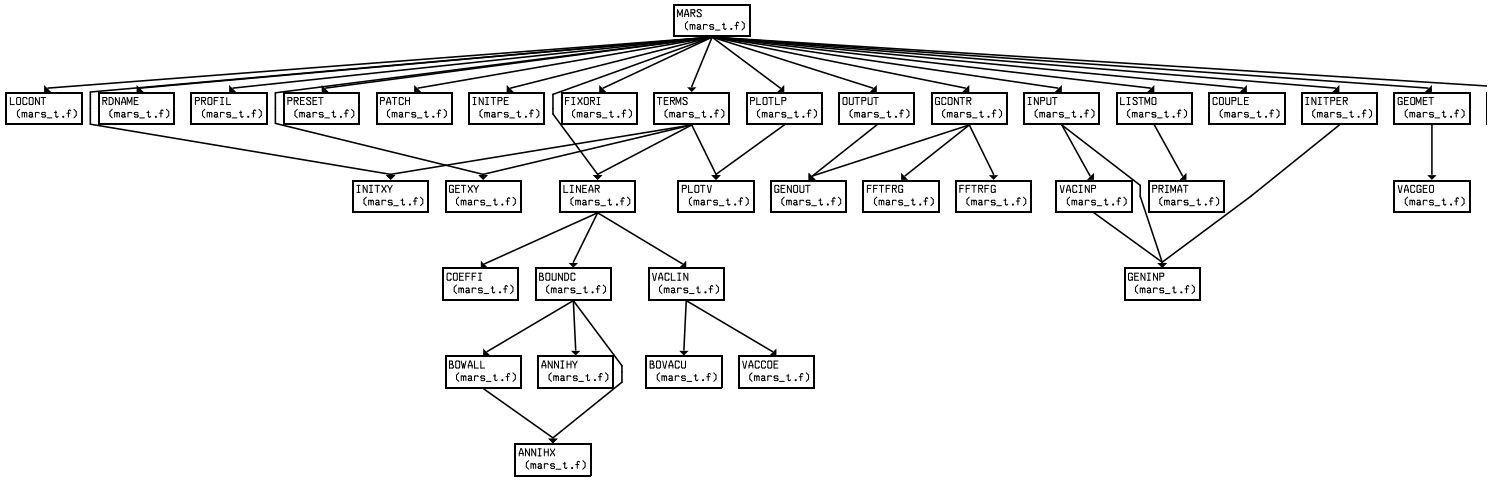
- ciascun nodo potenzialmente *multiprocessore*
- $\longrightarrow$ distribuzione *della memoria* e *del calcolo*
- $\longrightarrow$ le dimensioni del problema possono, in linea di principio, scalare con il numero di processori



I *processi* partecipanti hanno bisogno di *coordinazione* e di *comunicazione*.

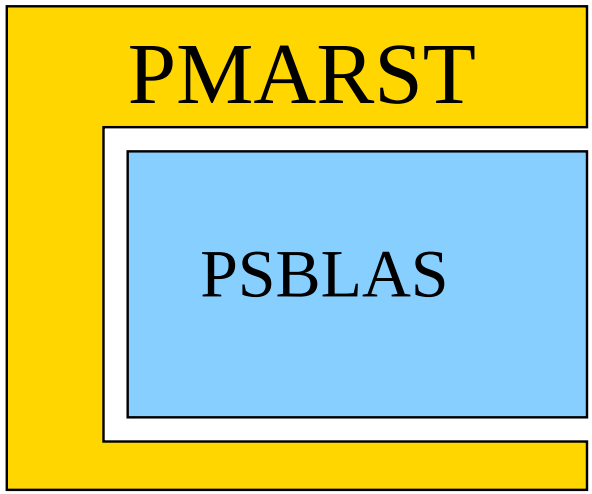
Riorganizzazione del codice

- Approccio bottom-up
- Studio *mirato* della struttura del codice sorgente (20000 righe)
- Uso e sviluppo di alcuni `tools` e `scripts`
- Porting del codice su GNU/Linux



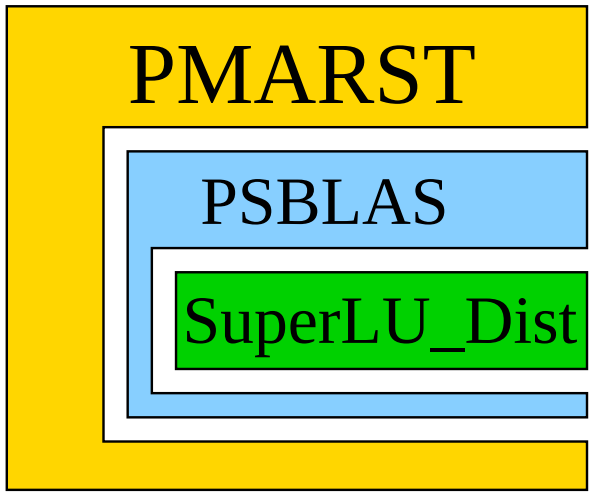
PSBLAS : Parallel Sparse BLAS

- Scritta in Fortran95, sfrutta MPI (Message Passing Interface)
- Si rifa alle ben note interfacce BLAS, LAPACK
- Per piattaforme UNIX : GNU/Linux, IBM AIX, SUN Solaris,..
- Pensato come *framework* per *metodi iterativi* (GMRES, BICGSTAB, CG, ...) di risoluzione
- Usato per problemi di *fluidodinamica computazionale*

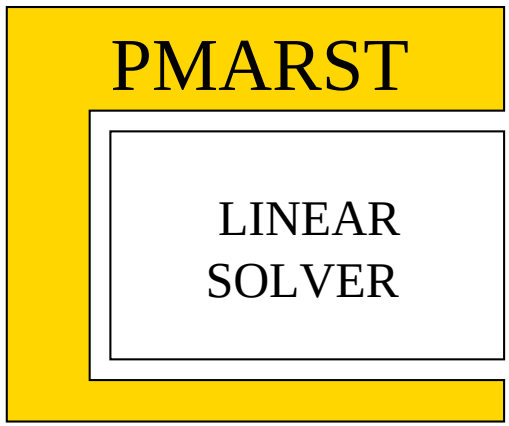


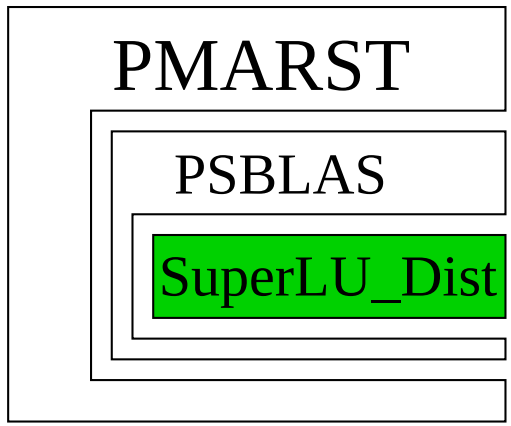
Un solutore *diretto* di grandi sistemi lineari *sparsi*.

- Scritta in ANSI C, sfrutta MPI
- Prodotto *Open Source*, del Lawrence Berkeley National Laboratory, USA
- È stato *integrato* con PSBLAS come *plugin*
- Per piattaforme UNIX
- Implementa *metodi diretti* di risoluzione



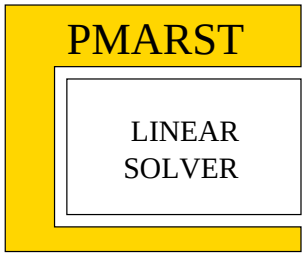
La bontà della *parallelizzazione* di MARST è valutata *separatamente* dall'*adeguatezza* di un *particolare linear solver* al problema di (P)MARST.

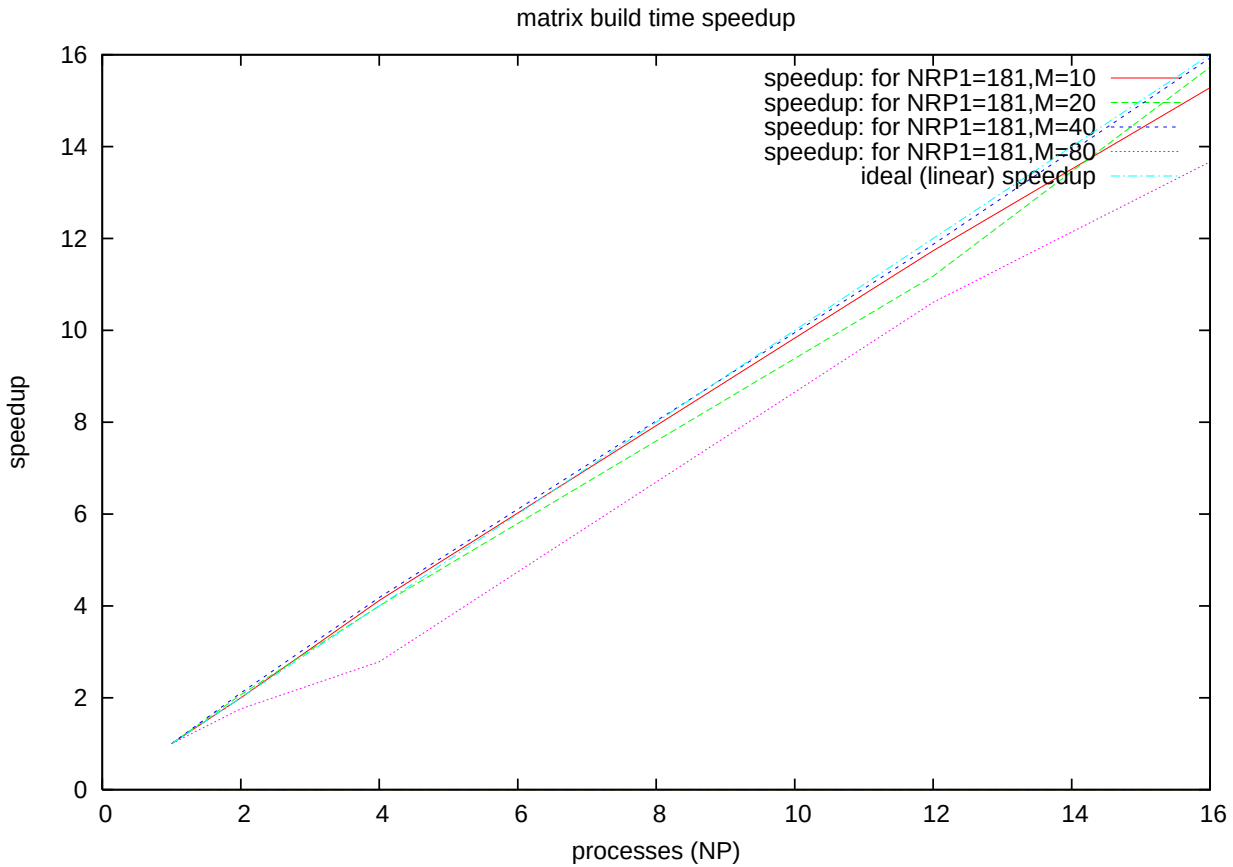




Efficienza della costruzione della matrice: *speedup*

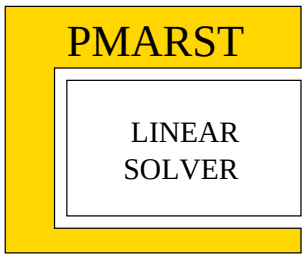
Lo *speedup* è *soddisfacente*

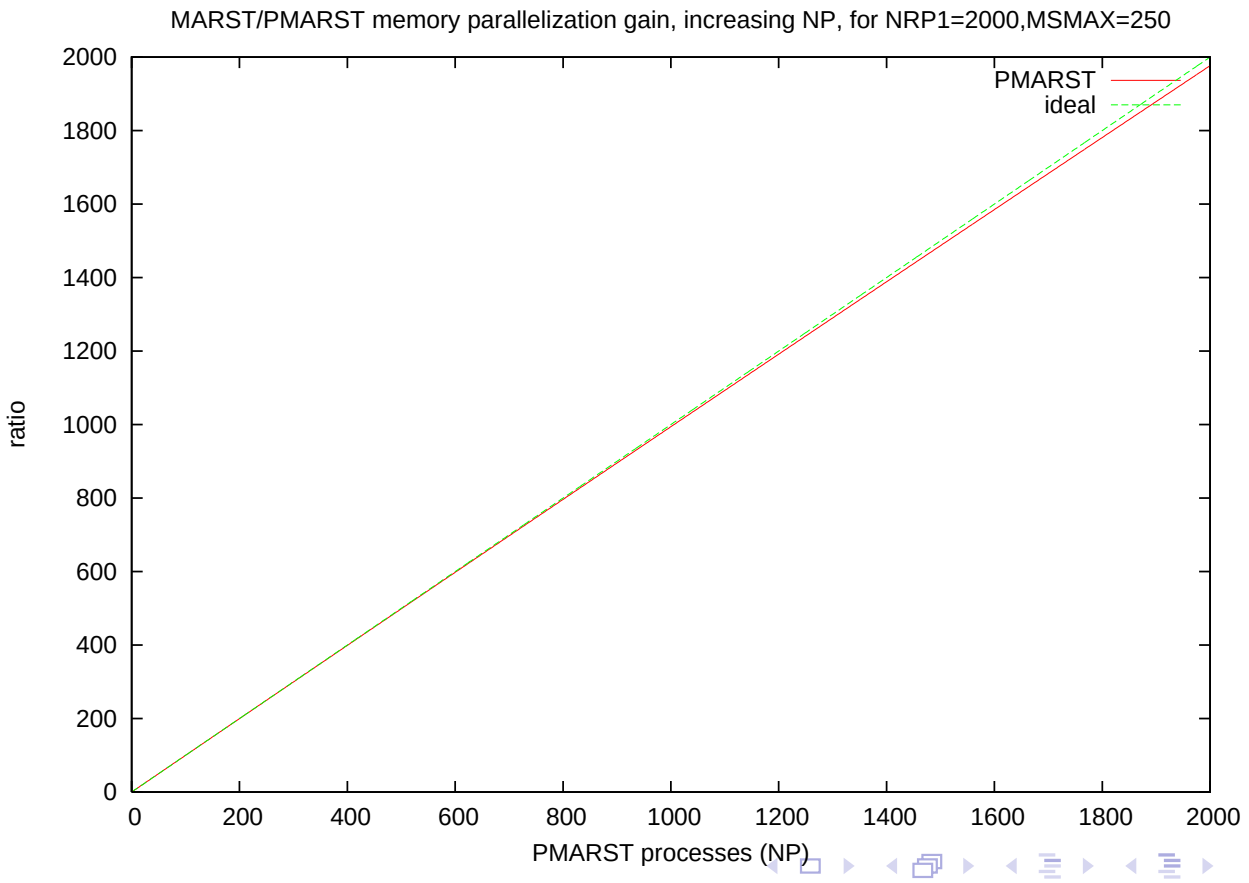




Efficienza della costruzione della matrice: *guadagno di memoria* (deterministico, calcolabile da dati in input)

Il *guadagno in memoria* è *quasi ideale*

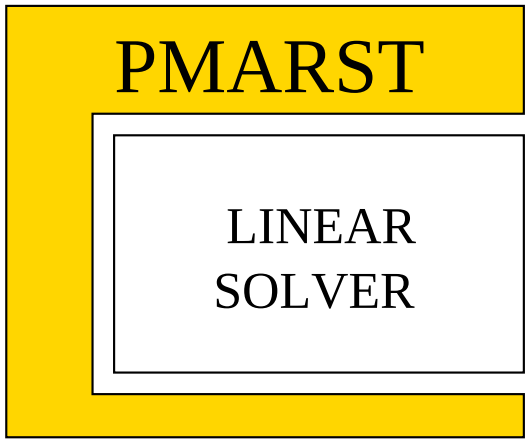




Conclusioni Efficienza *MARST Parallelo* (PMARST)

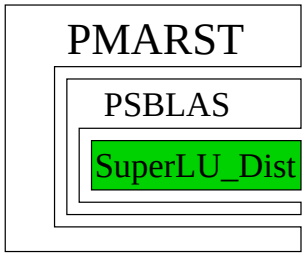
Risultati al crescere del grado di *parallelismo*:

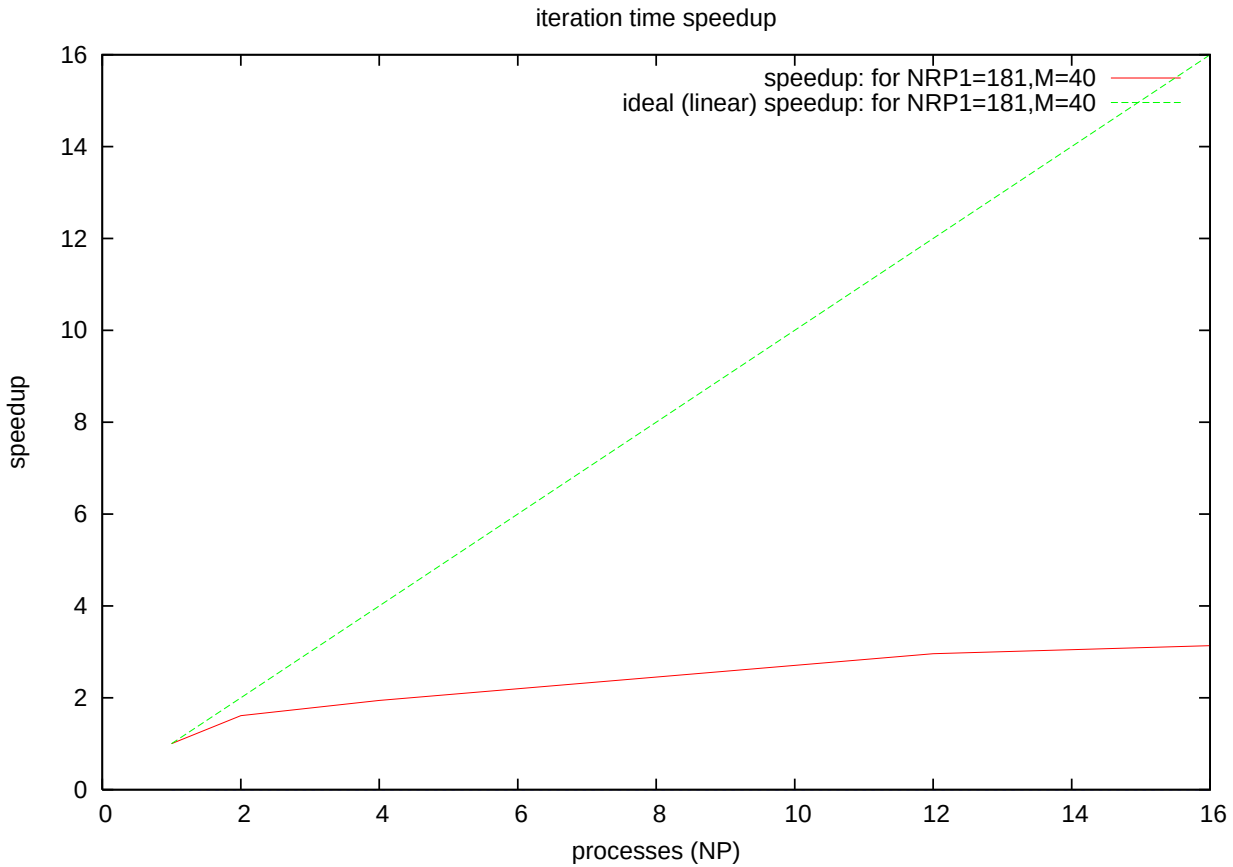
- soddisfacenti per *guadagno in memoria*
- soddisfacenti per *speedup* dei *tempi di risoluzione*



Speedup risoluzione sistema in SuperLU_Dist

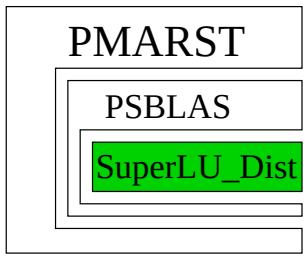
Speedup *scarso* ma *accettabile*

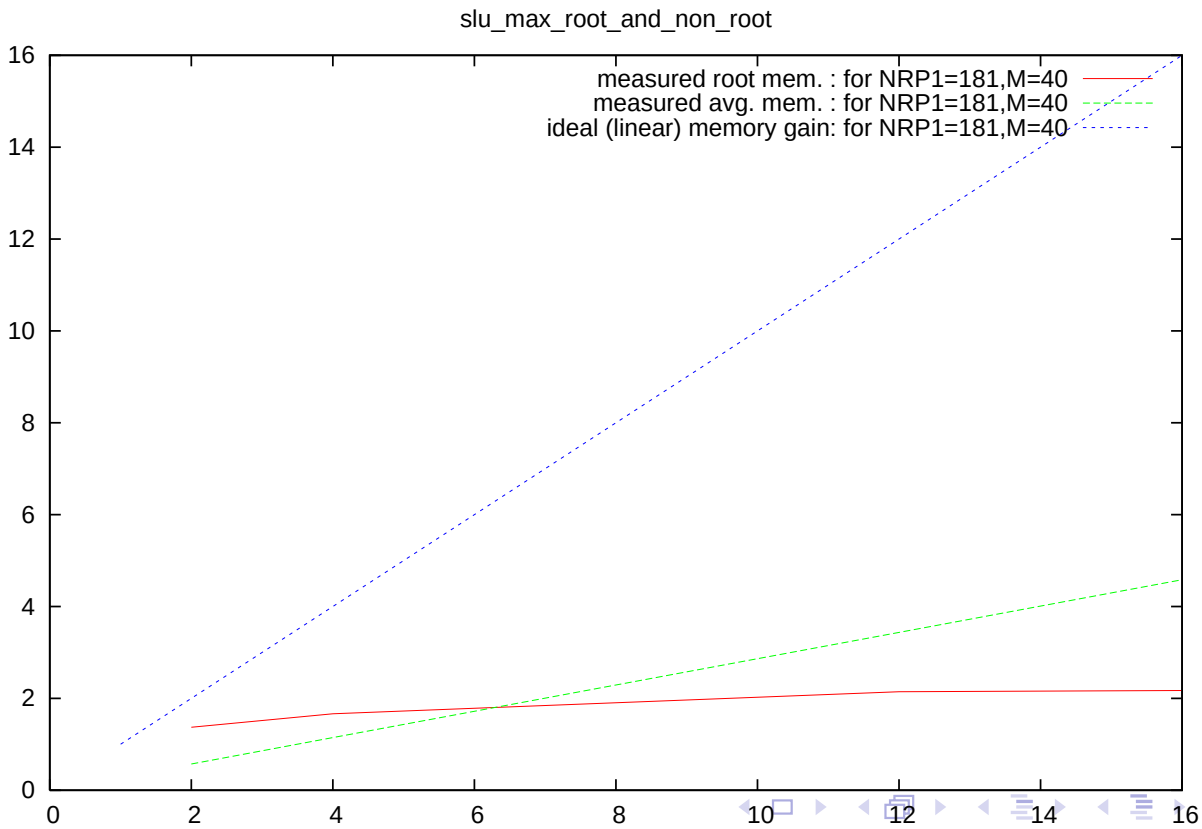




Efficienza della *risoluzione* del sistema dal punto di vista del *guadagno in memoria*

Scopriamo che il solutore SuperLU_Dist *satura* le risorse di memoria non *scalando* :

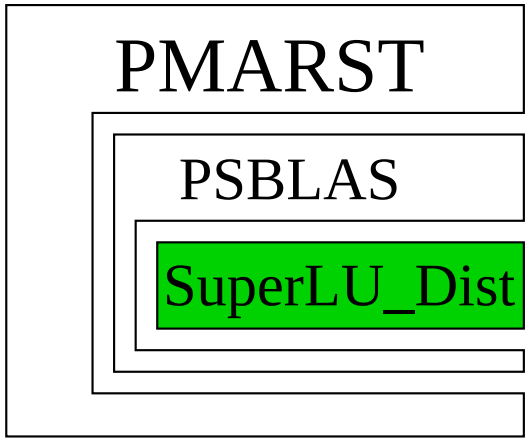




Conclusioni Adeguatezza SuperLU_Dist ai *sistemi lineari* di PMARST:

Risultati al crescere del grado di *parallelismo*:

- *accettabili* per *speedup* in questo *stadio* del lavoro
- ... ma *insoddisfacenti* per *guadagno in memoria* in prospettive *di aumento* del grado di parallelismo



Sviluppi lato PSBLAS

- Sviluppo di *precondizionatori* ad hoc.
- Testing metodi iterativi : a blocchi, dati problemi scalabilità *in alcune implementazioni*.
- Testing metodi diretti.
- Possibili modifiche a SuperLU_Dist e algoritmo di permutazione di Duff-Koster (tuttavia *arduo*).

Sviluppi lato PMARST

- Testing di solutori lineari è *necessario*.
- Interfacciamento con codice *particle-in-cell* per il conseguimento di un codice *ibrido*.

Conclusioni, in breve

- Il codice di MARST è ora parallelo e indipendente: In PMARST il sistema lineare in gioco viene generato in *parallelo*
- Resta la necessità di ricerca/sviluppo di un *linear solver* adatto al problema. Questo può essere un problema molto arduo.

Le seguenti slides sono di approfondimento!

PenGNUin

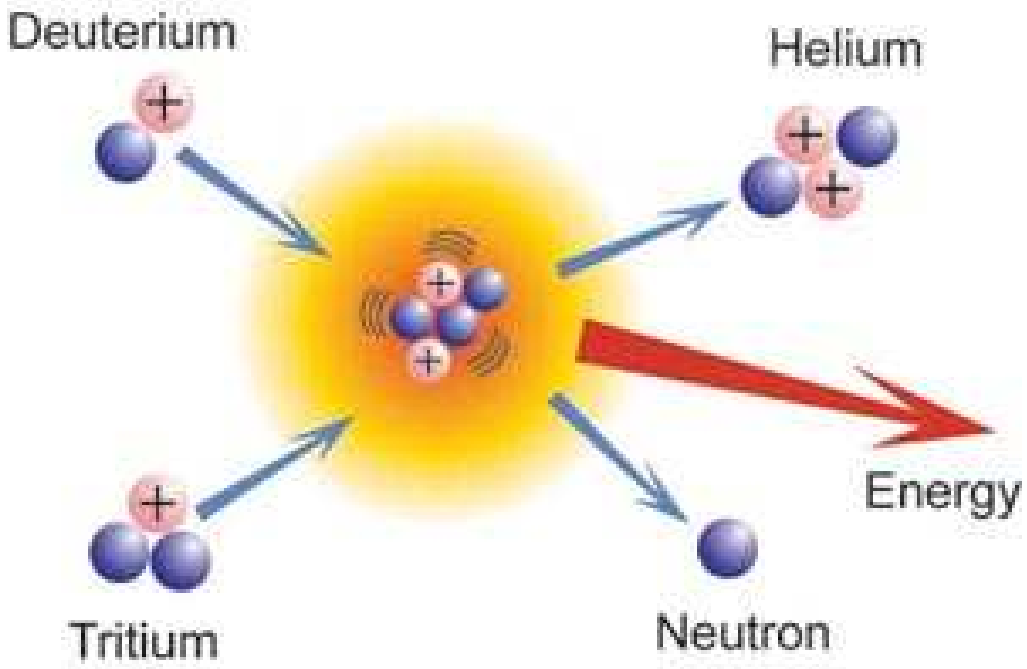
...e questa è *pubblicità virale* di :

`http://www.gnu.org/`

`http://www.linux.org/`

La fusione nucleare libera energia

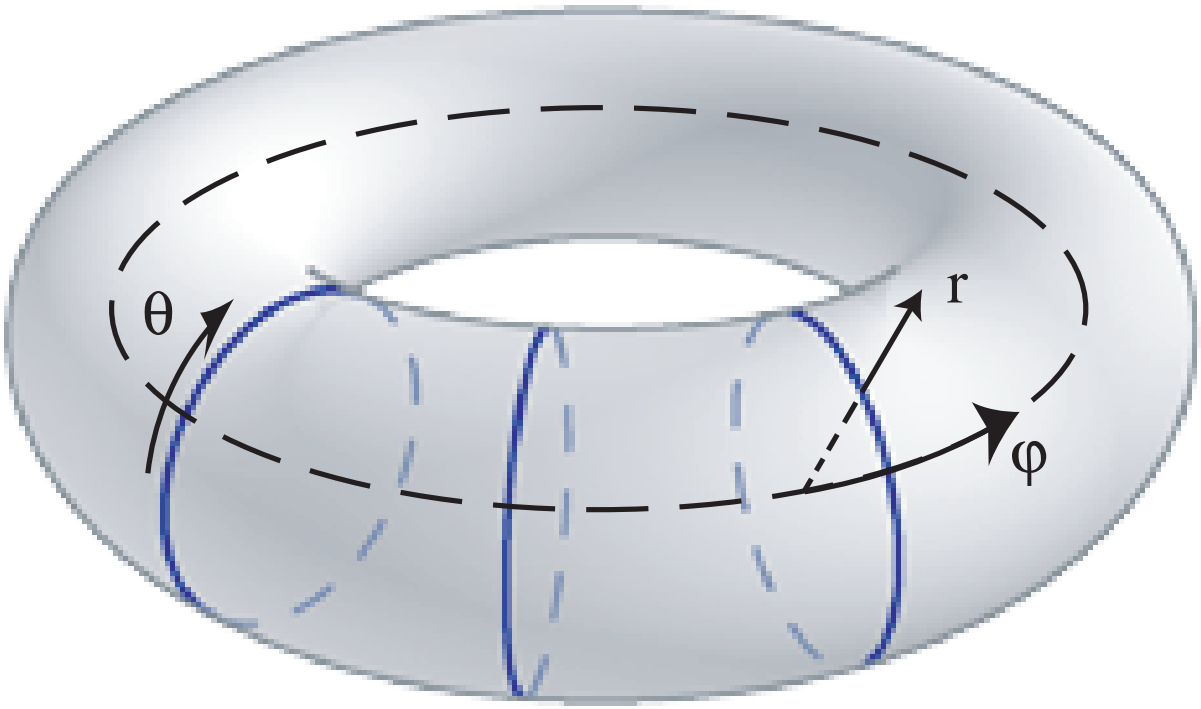
$$D + T \rightarrow He^4(3.5MeV) + n(14.1MeV)$$



Avviene continuamente nel sole e nelle stelle.

(TOroidalnaya KAmera MAgnitaya Katushka : **CAmera TOroidale a spira MAgnetica**)(A.Sacharov 196x)

Lunga tradizione di esperimenti : Jet(198x), Frascati(198x), FTU(199x), ITER(2008?),..



Un approccio sperimentale

- Usati alcuni tool di *reverse engineering*
- Impiego di *stubs* :
  - individuazione di PAMS
  - LAPACK *seriale*
  - PSBLAS *monoprocessore*
  - PSBLAS *parallelo*