ISSN/0393-3016

S. GIUSEPPONI, M. CELINO
Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Centro Ricerche Casaccia, Roma

G. BRACCO
Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Centro Ricerche Frascati, Roma

S. MIGLIORI
Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Sede Legale, Roma

PORTING E BENCHMARK IN AMBIENTE MULTIPIATTAFORMA ENEAGRID DI CODICI DI CALCOLO PER APPLICAZIONI SCIENTIFICHE, CON PARTICOLARE RIGUARDO PER QUELLI DEL SETTORE CHIMICA E FISICA DEI MATERIALI

(PARTE II)

RT/2017/25/ENEA

AGENZIA NAZIONALE PER LE NUOVE TECNOLOGIE,
L'ENERGIA E LO SVILUPPO ECONOMICO SOSTENIBILE

S. GIUSEPPONI, M. CELINO

Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Centro Ricerche Casaccia, Roma

G. BRACCO

Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Centro Ricerche Frascati, Roma

S. MIGLIORI

Dipartimento Tecnologie Energetiche
Divisione per lo Sviluppo Sistemi per l'Informatica e l'ICT
Sede Legale, Roma

PORTING E BENCHMARK IN AMBIENTE MULTIPIATTAFORMA ENEAGRID DI CODICI DI CALCOLO PER APPLICAZIONI SCIENTIFICHE, CON PARTICOLARE RIGUARDO PER QUELLI DEL SETTORE CHIMICA E FISICA DEI MATERIALI

(PARTE II)

RT/2017/25/ENEA

AGENZIA NAZIONALE PER LE NUOVE TECNOLOGIE,
L'ENERGIA E LO SVILUPPO ECONOMICO SOSTENIBILE

PORTING E BENCHMARK IN AMBIENTE MULTIPIATTAFORMA ENEAGRID DI CODICI DI CALCOLO PER APPLICAZIONI SCIENTIFICHE, CON PARTICOLARE RIGUARDO PER QUELLI DEL SETTORE CHIMICA E FISICA DEI MATERIALI

(PARTE II)

S. Giusepponi, M. Celino, G. Bracco, S. Migliori

Riassunto

In questo report descriviamo le principali tematiche affrontate nell'installazione in ambiente multipiattaforma ENEAGRID, di codici di calcolo per applicazioni nel settore della Scienza dei Materiali. Attualmente, l'architettura principalmente utilizzata è Linux x86_64, tuttavia, la descrizione di questo tipo di approccio multipiattaforma, mantiene una sua importanza perché illustra le potenzialità dell'architettura ENEAGRID che permettono flessibilità verso possibili evoluzioni future delle tecnologie per il calcolo scientifico.
Inizialmente, descriveremo il codice di dinamica molecolare CPMD e la sua installazione in ambiente multipiattaforma ENEAGRID. In seguito, analizzeremo le performance di calcolo fornite dall'infrastruttura computazionale CRESCO, ed infine, presenteremo alcuni dei risultati ottenuti grazie all'utilizzo di questo codice per lo studio di una interfaccia Magnesio – Idruro di Magnesio. L'analisi di questo tipo di interfaccia si colloca all'interno del tema di ricerca che ha come scopo quello di trovare nuove soluzioni di stoccaggio dell'idrogeno.

Parole chiave: Codici Scientifici, Calcolo ad Alte Prestazioni, Scienza dei Materiali Computazionale, Stoccaggio di Idrogeno.

Abstract

In this report, we describe the main issues addressed in the installation of computer codes for applications in the field of Materials Science in ENEAGRID. Currently, the main architecture used is Linux x86_64, however, the description of this type of multiplatform approach, remains important because it illustrates the potentialities of the ENEAGRID architecture that allow flexibility in future evolutions of scientific computing technologies.
Initially, we describe the molecular dynamics code CPMD and its installation on the multiplatform ENEAGRID environment. Later, the CPMD performance analysis on CRESCO infrastructure is given. Finally, we present results for the study of an interface between magnesium and magnesium hydride. The description of this interface is in the context of the research of new hydrogen storage solutions.

Key words: *Scientific Software, High Performance Computing, Computational Material Science, Hydrogen Storage.*

INDICE

Introduzione

Questo report descrive le principali tematiche affrontate relativamente all'attività di studio avente per oggetto: *Porting e benchmark in ambiente multipiattaforma ENEAGRID di codici di calcolo per applicazioni scientifiche, con particolare riguardo per quelli del settore Chimica e Fisica dei Materiali*, e rappresenta la seconda parte del rapporto tecnico RT-2017-12-ENEA[1].

Nella prima parte di questo report, verrà descritto il codice di dinamica molecolare CPMD e la sua installazione in ambiente multipiattaforma ENEAGRID. In seguito, verranno analizzate le performance di calcolo fornite dall'infrastruttura computazionale CRESCO, ed infine, saranno illustrati alcuni dei risultati ottenuti grazie all'utilizzo di questo codice per lo studio di una interfaccia Magnesio – Idruro di Magnesio. L'analisi di questo tipo di interfaccia si colloca all'interno del tema di ricerca che ha come scopo quello di trovare nuove soluzioni di stoccaggio dell'idrogeno.

1. Il codice di dinamica molecolare CPMD

Il codice CPMD (*Car-Parrinello Molecular Dynamics*) è un codice di simulazione di dinamica molecolare da primi principi, basato sulla teoria del funzionale densità con pseudopotenziali a norma conservata ed espansione in onde piane. Tale codice è disponibile nel sito `www.cpmd.org`, ed è possibile scaricarlo gratuitamente dopo essersi registrati. Dopo aver scaricato nella propria cartella di lavoro il file `cpmd-xxx.tar.gz` (dove `xxx` rappresenta la versione di CPMD), e averlo decompresso e scompattato, avremo una cartella nominata `CPMD-xxx`. Questa cartella è suddivisa nelle tre sottocartelle: `manual`, `PPLIBNEW` e `SOURCE`. Nella prima è disponibile il manuale di documentazione, nella seconda degli pseudopotanziali, mentre nell'ultima sono contenuti i file sorgente necessari per la compilazione e installazione di CPMD e la sottocartella `CONFIGURE` dove, sono disponibili i file di configurazione per differenti tipi di macchine di calcolo, di compilatori, di librerie scientifiche e di scelta di parallelizzazione. A scopo di esempio, di seguito riportiamo il file di configurazione fornito nel caso in cui si voglia installare la versione seriale di CPMD su un PC, compilando con `gfortran` e utilizzando staticamente le librerie matematiche `LAPACK` e `BLAS`. La descrizione di queste librerie è stata fornita nel rapporto RT-2017-12-ENEA[1].

```
=====================================================================
PC-GFORTRAN
#INFO#
#INFO# Configuration to build a serial cpmd executable for a linux
#INFO# pc using the GNU Fortran compiler.
#INFO#
```

[1] S.Giusepponi, M.Celino, G.Bracco, S.Migliori: *Porting e benchmark in ambiente multipiattaforma ENEAGRID di codici di calcolo per applicazioni scientifiche, con particolare riguardo per quelli del settore chimica e fisica dei materiali (Parte I)*. Documenti tecnici ENEA: RT-2017-12-ENEA, ISSN/0393-3016.

```
      IRAT=2
      CFLAGS='-c -O2 -Wall'
      CPP='/lib/cpp -P -C -traditional'
      CPPFLAGS='-D__Linux -D__PGI -D__GNU -DFFT_DEFAULT'
      FFLAGS='-c -fdefault-real-8 -O2 -fcray-pointer'
      LFLAGS='-llapack -lblas -static'
      FFLAGS_GROMOS='  $(FFLAGS)'
      if [ $debug ]; then
        FC='gfortran -g'
        CC='gcc -g'
        LD='gfortran -g'
      else
        FC='gfortran '
        CC='gcc '
        LD='gfortran '
      fi
=====================================================================
```

Per la compilazione, si dovrà editare il file di configurazione più simile alla propria macchina di calcolo, apportare le dovute correzioni e salvarlo ad esempio come `MY-PLATFORM`, nella directory `CONFIGURE` e digitare dalla cartella `SOURCE`:

```
./mkconfig.sh MY-PLATFORM > Makefile
```

questo commando produrrà il `Makefile` che verrà eseguito digitando `make` sempre nella directory `SOURCE`. Alla fine della compilazione verrà creato il file eseguibile `cpmd.x`.

2. Installazione di CPMD in ENEA-GRID

Sulla base delle indicazioni fornite nel paragrafo precedente, abbiamo provveduto all'istallazione del codice di dinamica molecolare CPMD su alcune delle risorse di calcolo disponibili presso i centri di ricerca ENEA. In particolare si sono scelte le macchine IBM e l'infrastruttura di calcolo CRESCO (`www.cresco.enea.it`). Per un elenco completo delle risorse di calcolo disponibili presso i centri ENEA si consulti la pagina web:
`http://www.afs.enea.it/project/eneagrid/Resources/List.html`.

2.1. Installazione su macchine IBM

Nel centro di calcolo dell'ENEA di Frascati sono disponibili due tipi di macchine IBM. Sono presenti 4 nodi SP4 ciascuno costituito da 32 processori Power4 con sistema operativo `AIX 5.2.4`, e 13 nodi SP5 con sistema operativo `AIX 5.3.3` per un totale di 256 processori Power5, ripartiti in 12 nodi con 16 processori ciascuno, più un nodo con 64 processori. Per entrambe le configurazioni (SP4 ed SP5) abbiamo utilizzato i software proprietari IBM; ovvero il compilatore `xlf` (*XL Fortran Enterprise Edition V10.1*), le librerie matematiche `ESSL` (*Engineering and Scientific Subruotine Library*). Va osservato che le librerie `ESSL` contengono tutte le subroutine `BLAS`, ma mancano alcune subroutine delle librerie `LAPACK`, per tale ragione queste subroutine mancanti sono state compilate separatamente e chiamante rispettivamente `lapack_SP4-32_xlf_essl` e `lapack_SP5-64_xlf_essl`.

Per il primo tipo di macchina si sono compilate quattro versioni di CPMD; installando così, sia la versione parallela `MPI`, sia quella mista `MPI-SMP` e compilando sia a 32 bit cha a 64 bit. Di seguito riportiamo i file di configurazione relativi a queste quattro versioni di CPMD, sulla base dei quali mediante il file di script `mkconfig.sh` verranno prodotti i `Makefile` e quindi l'installazione vera e propria.

```
=============================================================
SP4-32-MPI
#INFO#
#INFO# p690 MPI only 32-bit - safe version
#INFO#
      IRAT=2
      FFLAGS='-qmaxmem=32768 -qtune=pwr4 -qarch=pwr4'
      FFLAGS_GROMOS='-qarch=pwr4 -qdpc'
      LFLAGS='-L/afs/enea.it/fra/user/giuseps/LIB -llapack_SP4-
32_xlf_essl -lessl -llapack_SP4-32_xlf_essl -bbinder:/usr/lib/bind -
bmaxdata:2048000000 -qarch=pwr4'
      CFLAGS='-O3 -qstrict -qarch=pwr4'
      CPP='/usr/ccs/lib/cpp -P'
      CPPFLAGS='-D__IBM -DPARALLEL=PARALLEL -DFFT_ESSL'
      AR='/usr/bin/ar'
      RANLIB='/usr/bin/ranlib'
```

```
     if [ $debug ]; then
          CC='xlc_r -c -g -C -qflttrap'
          FC='mpxlf_r -c -g -C -qflttrap'
          LD='mpxlf_r -g -C -qflttrap'
     else
          CC='xlc_r -c -O3 -qstrict'
          FC='mpxlf_r -c -O3 -qstrict'
          LD='mpxlf_r -O3 -qstrict'
     fi
=====================================================================
=====================================================================
SP4-32-MPI+SMP
#INFO#
#INFO# p690 MPI+SMP 32-bit - safe version
#INFO#
     IRAT=2
     FFLAGS='-qmaxmem=32768 -qtune=pwr4 -qarch=pwr4 -qsmp=omp'
     FFLAGS_GROMOS='-qarch=pwr4 -qdpc'
     LFLAGS='-L/afs/enea.it/fra/user/giuseps/LIB -llapack_SP4-
32_xlf_essl -lesslsmp -llapack_SP4-32_xlf_essl
-bbinder:/usr/lib/bind -bmaxdata:2048000000 -qarch=pwr4 -qsmp=omp'
     CFLAGS='-qarch=pwr4'
     CPP='/usr/ccs/lib/cpp -P'
     CPPFLAGS='-D__IBM -DPARALLEL=PARALLEL -DFFT_ESSL'
     NOOPT_FLAG='-O0'
     NOOPT_OBJS='control.o memory.o freem.o azzero.o'
     AR='/usr/bin/ar'
     RANLIB='/usr/bin/ranlib'
     if [ $debug ]; then
          CC='xlc_r -c -g -C -qflttrap'
          FC='mpxlf_r -c -g -C -qflttrap'
          LD='mpxlf_r -g -C -qflttrap'
     else
          CC='xlc_r -c -O3 -qstrict'
          FC='mpxlf_r -c -O3 -qstrict'
          LD='mpxlf_r -O3 -qstrict'
     fi
=====================================================================
```

```
=====================================================================
SP4-64-MPI
#INFO#
#INFO# p690 MPI only 64-bit - safe version - use 64bit compiled #INFO#
Lapack
     IRAT=2
     FFLAGS='-O3 -qstrict -qmaxmem=32768 -qtune=pwr4 -qarch=pwr4
-q64'
     FFLAGS_GROMOS='-c -O3 -qarch=pwr4 -qdpc'
     LFLAGS='-L/afs/enea.it/fra/user/giuseps/LIB -llapack_SP4-
64_xlf_essl -lessl -llapack_SP4-64_xlf_essl -q64
-bbinder:/usr/lib/bind -bmaxdata:32768000000 -qarch=pwr4'
     CFLAGS='-O3 -qstrict -qarch=pwr4 -q64'
     CPP='/usr/ccs/lib/cpp -P'
     CPPFLAGS='-D__IBM -DPOINTER8 -DPARALLEL=PARALLEL -DFFT_ESSL'
     NOOPT_FLAG='-O0 -q64'
     NOOPT_OBJS='control.o memory.o freem.o'
     AR='/usr/bin/ar'
     RANLIB='/usr/bin/ranlib'
     if [ $debug ]; then
          CC='xlc_r -c -g -C -qflttrap'
          FC='mpxlf_r -c -g -C -qflttrap'
          LD='mpxlf_r -g -C -qflttrap'
     else
          CC='xlc_r -c -O3 -qstrict'
          FC='mpxlf_r -c -O3 -qstrict'
          LD='mpxlf_r -O3 -qstrict'
     fi
=====================================================================

=====================================================================
SP4-64-MPI+SMP
#INFO#
#INFO# p690 MPI+SMP 64-bit - safe version - use 64 bit compiled #INFO#
lapack
     IRAT=2
     FFLAGS='-q64 -qmaxmem=32768 -qtune=pwr4 -qarch=pwr4
-qsmp=omp'
```

```
      FFLAGS_GROMOS='-qarch=pwr4 -qdpc'
      LFLAGS='-L/afs/enea.it/fra/user/giuseps/LIB -llapack_SP4-
64_xlf_essl -lesslsmp -llapack_SP4-64_xlf_essl -q64
-bbinder:/usr/lib/bind -bmaxdata:32768000000 -qarch=pwr4
-qsmp=omp'
      CFLAGS='-qarch=pwr4 -q64'
      CPP='/usr/ccs/lib/cpp -P'
      CPPFLAGS='-D__IBM -DPOINTER8 -DPARALLEL=PARALLEL -DFFT_ESSL'
      NOOPT_FLAG='-O0 -q64 '
      NOOPT_OBJS='control.o memory.o freem.o azzero.o '
      AR='/usr/bin/ar'
      RANLIB='/usr/bin/ranlib'
      if [ $debug ]; then
            CC='xlc_r -c -g -C -qflttrap'
            FC='mpxlf_r -c -g -C -qflttrap'
            LD='mpxlf_r -g -C -qflttrap'
      else
            CC='xlc_r -c -O3 -qstrict'
            FC='mpxlf_r -c -O3 -qstrict'
            LD='mpxlf_r -O3 -qstrict'
      fi
=====================================================================
```

Per l'altro tipo di macchina IBM, si sono scelte sempre quattro tipi di installazioni, anche se in questi casi tutte compilate a 64 bit. Le versioni sono: seriale, parallela con `MPI`, parallela con `SMP` e parallela mista `MPI-SMP`. Di seguito riportiamo i vari file di configurazione.

```
=====================================================================
SP5-SERIALE
#INFO#
#INFO# IBM-PWR5-AIX-serial-64bit-essl
#INFO# you need lapack and essl 64 bit libraries
#INFO#
      IRAT=2
      FFLAGS='-q64 -qmaxmem=32768 -qtune=pwr5 -qarch=pwr5'
      FFLAGS_GROMOS='-qarch=pwr5 -qdpc'
      LFLAGS='-q64 -L/afs/enea.it/fra/user/giuseps/LIB
-llapack_SP5-64_xlf_essl -lessl -qarch=pwr5'
      CFLAGS='-q64 -qarch=pwr5'
```

```
     CPP='/usr/ccs/lib/cpp -P'
     CPPFLAGS='-D__IBM -DLAPACK -DFFT_ESSL -DMALLOC8 -DPOINTER8'
     NOOPT_FLAG='-O0 -q64'
     NOOPT_OBJS='control.o memory.o freem.o'
     AR='/usr/bin/ar ruv'
     RANLIB='/usr/bin/ranlib'
     if [ $debug ]; then
          CC='xlc_r -c -g -C -qflttrap'
          FC='xlf_r -c -g -C -qflttrap'
          LD='xlf_r -g -C -qflttrap'
     else
          CC='xlc_r -c -O3 -qstrict'
          FC='xlf_r -c -O3 -qstrict'
          LD='xlf_r  -O3 -qstrict'
     fi
=======================================================================

=======================================================================
SP5-64-MPI
#INFO#
#INFO# IBM-PWR5-AIX-MPI-64bit-essl
#INFO# you need lapack and essl 64 bit libraries
#INFO#
     IRAT=2
     FFLAGS='-q64 -qmaxmem=32768 -qtune=pwr5 -qarch=pwr5'
     FFLAGS_GROMOS='-qarch=pwr5 -qdpc'
     LFLAGS='-q64 -L/afs/enea.it/fra/user/giuseps/LIB
-llapack_SP5-64_xlf_essl -lessl -qarch=pwr5'
     CFLAGS='-q64 -qarch=pwr5'
     CPP='/usr/ccs/lib/cpp -P'
     CPPFLAGS='-D__IBM -DLAPACK -DFFT_ESSL -DMALLOC8 -DPOINTER8
-DPARALLEL=PARALLEL -DMP_LIBRARY=__MPI'
     NOOPT_FLAG='-O0 -q64'
     NOOPT_OBJS='control.o memory.o freem.o'
     AR='/usr/bin/ar ruv'
     RANLIB='/usr/bin/ranlib'
     if [ $debug ]; then
          CC='xlc_r -c -g -C -qflttrap'
```

```
          FC='mpxlf_r -c -g -C -qflttrap'
          LD='mpxlf_r -g -C -qflttrap'
     else
          CC='xlc_r -c -O3 -qstrict'
          FC='mpxlf_r -c -O3 -qstrict'
          LD='mpxlf_r  -O3 -qstrict'
     fi
=====================================================================

=====================================================================
SP5-64-SMP
#INFO#
#INFO# IBM-PWR5-AIX-smp-64bit-essl
#INFO# you need lapack and essl 64 bit libraries
#INFO#
     IRAT=2
     FFLAGS='-q64 -qmaxmem=32768 -qtune=pwr5 -qarch=pwr5
-qsmp=omp'
     FFLAGS_GROMOS='-O3 -qarch=pwr5 -qdpc'
     LFLAGS='-q64 -L/afs/enea.it/fra/user/giuseps/LIB
-llapack_SP5-64_xlf_essl -lesslsmp -qarch=pwr5 -qsmp=omp'
     CFLAGS='-q64 -O3 -qstrict -qarch=pwr5'
     CPP='/usr/ccs/lib/cpp -P'
     CPPFLAGS='-D__IBM -DLAPACK -DFFT_ESSL -DMALLOC8 -DPOINTER8'
     NOOPT_FLAG='-O0 -q64'
     NOOPT_OBJS='control.o memory.o freem.o'
     AR='/usr/bin/ar ruv'
     RANLIB='/usr/bin/ranlib'
     if [ $debug ]; then
          CC='xlc_r -c -g -C -qflttrap'
          FC='xlf_r -c -g -C -qflttrap'
          LD='xlf_r -g -C -qflttrap'
     else
          CC='xlc_r -c -O3 -qstrict'
          FC='xlf_r -c -O3 -qstrict'
          LD='xlf_r  -O3 -qstrict'
     fi
=====================================================================
```

```
=================================================================
SP5-64-MPI-SMP
#INFO#
#INFO# IBM-PWR5-AIX-MPI-smp-64bit-essl
#INFO# you need lapack and essl 64 bit libraries
#INFO#
     IRAT=2
     FFLAGS='-q64 -qmaxmem=32768 -qtune=pwr5 -qarch=pwr5
-qsmp=omp'
     FFLAGS_GROMOS='-qarch=pwr5 -qdpc'
     LFLAGS='-q64 -L/afs/enea.it/fra/user/giuseps/LIB
-llapack_SP5-64xlfessl -lesslsmp -qarch=pwr5 -qsmp=omp'
     CFLAGS='-q64 -qarch=pwr5'
     CPP='/usr/ccs/lib/cpp -P'
     CPPFLAGS='-D__IBM -DLAPACK -DFFT_ESSL -DMALLOC8 -DPOINTER8
-DPARALLEL -DMP_LIBRARY=__MPI'
     NOOPT_FLAG='-O0 -q64'
     NOOPT_OBJS='control.o memory.o freem.o'
     AR='/usr/bin/ar ruv'
     RANLIB='/usr/bin/ranlib'
     if [ $debug ]; then
          CC='xlc_r -c -g -C -qflttrap'
          FC='mpxlf_r -c -g -C -qflttrap'
          LD='mpxlf_r -g -C -qflttrap'
     else
          CC='xlc_r -c -O3 -qstrict'
          FC='mpxlf_r -c -O3 -qstrict'
          LD='mpxlf_r  -O3 -qstrict'
     fi
=================================================================
```

Da varie prove abbiamo visto che per questo tipo di macchine e per il sistema che abbiamo intenzione di studiare (interfaccia magnesio-idruro di magnesio), le versioni parallele che risultano scalare meglio con il numero di processori sono quelle che prevedono la sola parallelizzazione `MPI`. Nel rapporto tecnico RT-2017-12-ENEA è stata fatta una analisi di questi aspetti, in particolare sono state messe in evidenza le ragioni per le quali si raggiunge una saturazione della scalabilità del sistema con l'aumentare del numero di processori impiegato nelle simulazioni. Sulla base di queste conoscenze abbiamo provveduto all'installazione di questo codice di dinamica molecolare sull'infrastruttura di calcolo dell'ENEA chiamata CRESCO: *Centro*

computazionale di RicErca sui Sistemi COmplessi (per ulteriori informazioni si consulti il sito `www.cresco.enea.it`).

2.2. Istallazione sulla struttura di calcolo CRESCO

L'infrastruttura di calcolo CRESCO, che è situata presso il centro di ricerca ENEA di Portici, rappresenta una delle principali strutture computazionali presenti in Italia e classificandosi al 126° posto nella classifica TOP500 di giugno 2008 (`www.top500.org`). Il sistema CRESCO risulta suddiviso in tre sezioni: la prima, dedicata a codici che richiedono un gran utilizzo di memoria RAM, è costituita da 42 nodi SMP IBM x3850-M2 ciascuno con 4 processori Intel Xeon Quad-Core Tigerton E7330 per un totale di 672 core. La seconda sezione, dedicata a codici ad alto parallelismo, è costituita da 256 nodi blades IBM HS21 ciascuno con 2 processori Intel Xeon Quad-Core Clovertown E5345 per un totale di 2048 core. Infine l'ultima sezione, dedicata a scopi particolari, è costituita da 18 nodi sui quali sono presenti processori di diverso tipo. Poiché CPMD rappresenta un codice ad alto parallelismo, e quindi destinato ad essere utilizzato principalmente sulla seconda sezione di CRESCO, abbiamo cercato di compilarlo in modo tale da essere il più performante possibile su tale sezione. Per tale ragione abbiamo utilizzato il compilatore Intel `ifort` (*Intel Fortran Compiler, versione 10.1*) e librerie scientifiche Intel `MKL` (*Math Kernel Library, versione 10.1*), inoltre, abbiamo utilizzato le librerie `MPI openmpi_intel-1.2.5` compilate anche esse con `ifort`. Di seguito riportiamo i file di configurazione relativi alle istallazioni parallela e seriale di CPMD, sulla base dei quali mediante il file di script `mkconfig.sh` verranno prodotti i `Makefile`. Va osservato che abbiamo scelto di fare il link statico alle librerie scientifiche `MKL`.

```
=====================================================================
CRESCO-MPI-MKL
#INFO#
#INFO# Configuration file to build a parallel cpmd executable for
#INFO# a linux machine with an AMD64/EM64T cpu
#INFO# (Opteron/AthlonFX/Athlon64/Xeon-EM64T) using
#INFO# the Intel Fortran Compiler with EM64T extensions.
#INFO#
      IRAT=2
     CFLAGS='-O2 -Wall -m64'
     CPP='/lib/cpp -P -C -traditional'
      CPPFLAGS='-D__Linux -D__PGI -DFFT_DEFAULT -DPOINTER8
-DINTEL_MKL -DLINUX_IFC -DPARALLEL -DMYRINET'
     FFLAGS='-pc64 -O2 -unroll -march=pentium3 -mtune=core2'
     LFLAGS='-L/afs/enea.it/software/inteldev/x86_64_rhel5/
mkl/10.0.1.014/lib/em64t -I/afs/enea.it/software/inteldev/
x86_64_rhel5/mkl/10.0.1.014/include -Wl,--start-group
```

```
/afs/enea.it/software/inteldev/x86_64_rhel5/mkl/10.0.1.014/lib/em64t/lib
mkl_intel_lp64.a
/afs/enea.it/software/inteldev/x86_64_rhel5/mkl/10.0.1.014/lib/em64t/lib
mkl_sequential.a
/afs/enea.it/software/inteldev/x86_64_rhel5/mkl/10.0.1.014/lib/em64t/lib
mkl_core.a -Wl,--end-group'
     FFLAGS_GROMOS='-Dgood_luck $(FFLAGS)'
     if [ $debug ]; then
          FC='mpif77 -c -g'
          CC='mpicc -g -Wall -m64'
          LD='mpif77 -g'
     else
          FC='mpif77 -c'
          CC='mpicc'
          LD='mpif77 -i-static '
     fi
=====================================================================

=====================================================================
CRESCO-MKL-SERIALE
#INFO#
#INFO# Configuration file to build a serial cpmd executable for a #INFO#
linux machine with an AMD64/EM64T cpu
#INFO# (Opteron/AthlonFX/Athlon64/Xeon-EM64T) using
#INFO# the Intel Fortran Compiler with EM64T extensions.
#INFO#
     IRAT=2
     CFLAGS='-O2 -Wall -m64'
     CPP='/lib/cpp -P -C -traditional'
     CPPFLAGS='-D__Linux -D__PGI -DFFT_DEFAULT -DPOINTER8
-DLINUX_IFC -DINTEL_MKL'
     FFLAGS='-pc64 -O2 -unroll -march=pentium3 -mtune=core2'
     LFLAGS=' -L/afs/enea.it/software/inteldev/x86_64_rhel5/
mkl/10.0.1.014/lib/em64t -I/afs/enea.it/software/inteldev/
x86_64_rhel5/mkl/10.0.1.014/include -Wl,--start-group
/afs/enea.it/software/inteldev/x86_64_rhel5/mkl/10.0.1.014/lib/em64t/lib
mkl_intel_lp64.a
/afs/enea.it/software/inteldev/x86_64_rhel5/mkl/10.0.1.014/lib/em64t/lib
```

```
mkl_sequential.a
/afs/enea.it/software/inteldev/x86_64_rhel5/mkl/10.0.1.014/lib/em64t/lib
mkl_core.a -Wl,--end-group '
     FFLAGS_GROMOS='-Dgood_luck $(FFLAGS)'
     if [ $debug ]; then
          FC='ifort -c -g'
          CC='gcc -g -Wall -m64'
          LD='ifort -g'
     else
          FC='ifort -c'
          CC='gcc'
          LD='ifort -i-static'
     fi
=====================================================================
```

3. Benchmark e ottimizzazione di CPMD sulla struttura di supercalcolo CRESCO

In questa sezione presentiamo i risultati dello studio di scalabilità del codice al variare con il numero di core impiegati nelle simulazioni. Per confrontare questi risultai con quelli presentati nel rapporto tecnico RT-2017-12-ENEA, nel quale veniva analizzata la scalabilità sulle macchine IBM SP5, consideriamo lo stesso tipo di problema, e cioè, consideriamo il sistema costituito da una interfaccia idruro di magnesio (MgH_2) – magnesio (Mg), così come mostrato nella Figura 1. Questa scelta è condizionata dal fatto che questa problematica è l'oggetto del nostro tema di ricerca.

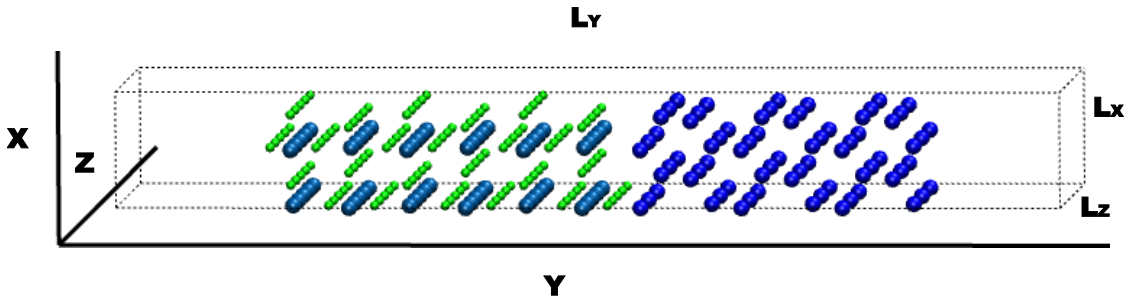

Figura 1. Interfaccia Idruro di Magnesio (MgH_2) – Magnesio (Mg).

Tale sistema è costituito da 132 atomi di magnesio e 120 atomi di idrogeno, le dimensioni della cella di simulazione sono L_x= 6.21 Å; L_y= 50.3 Å; L_z= 15.1 Å. Utilizzando CPMD per lo studio di questo sistema, abbiamo visto che il codice sceglie la *mesh* 72x576x180 per il calcolo delle trasformate di Fourier 3D, e ripartisce il carico di lavoro tra i vari processi seguendo il primo indice. Di seguito nella Tabella 1, riportiamo al variare del numero di core impiegati, i tempi (in secondi s) di esecuzione relativi a 300 passi di dinamica molecolare con *timestep* = 2 a.u. e *cutoff* = 80 a.u.

Dai tempi medi di esecuzione per ciascun passo di dinamica molecolare, si vede che ha senso incrementare fino a 72 il numero di core utilizzati. Tale numero coincide con quello del primo indice della *mesh*. Per diminuire i tempi di esecuzione aumentando il numero dei processi, abbiamo sfruttato l'opzione di CPMD che permette di selezionare il numero di `TASKGROUPS`. Di seguito riportiamo le tabelle con i tempi di esecuzione nel caso in cui si ponga `TASKGROUPS`=2 (Tabella 2), ovvero, `TASKGROUPS`=3 (Tabella 3).

Numero core	Initialization time	Time step medio	Cpu time	Elapsed time
1	391.80	388.06	116822.19	119217.65
2	191.68	230.13	69247.33	70012.51
4	119.78	123.07	37052.02	37748.64
8	91.47	93.22	28069.97	29041.33

16	49.58	48.36	14568.77	15603.22
32	30.28	28.91	8715.34	9634.20
36	30.40	27.61	8330.02	9378.95
48	24.54	22.48	6782.97	7970.06
64	22.23	18.44	5668.94	7805.27
72	15.29	11.08	3352.75	4274.51
144	15.54	10.09	3063.01	4208.14
216	20.49	11.21	3404.57	4399.53

Tabella 1. Tempi di esecuzione (s) per 300 passi di dinamica molecolare.

Numero core	Initialization time	Time step medio	Cpu time	Elapsed time
2	192.19	256.69	77216.58	78266.87
4	133.95	153.38	46161.67	47267.55
8	77.21	77.38	23304.42	24338.87
16	51.62	50.35	15167.97	16324.00
32	28.72	27.67	8343.39	11523.97
36	27.98	26.57	8010.56	8994.93
48	19.76	17.96	5420.00	6821.54
64	18.97	16.49	4978.73	6260.42
72	19.28	14.75	4466.71	5642.26
144	11.31	6.25	1907.10	2833.25
216	14.48	5.93	1812.35	2873.37
288	24.72	7.36	2259.99	2934.06

Tabella 2. Tempi di esecuzione (s) per 300 passi di dinamica molecolare avendo fissato `TASKGROUPS`=2.

Numero core	Initialization time	Time step medio	Cpu time	Elapsed time
3	163.18	167.89	50541.37	51695.4
6	123.86	121.09	36462.02	37147.2
18	52.07	50.51	15217.58	16179.35
36	29.09	27.46	8280.22	9332.87
48	23.53	20.58	6209.13	7445.38
72	15.11	11.97	3624.00	4575.13
144	14.49	9.59	2910.06	3834.64
216	14.3	4.85	1491.46	2405.44
288	27.39	5.90	1818.17	2690.76

Tabella 3. Tempi di esecuzione (s) per 300 passi di dinamica molecolare avendo fissato `TASKGROUPS`=3.

Operando ulteriormente sia sulle opzioni del codice CPMD sia sfruttando le potenzialità della struttura di supercalcolo CRESCO, è possibile ridurre ulteriormente i tempi di esecuzione delle simulazioni. In particolare, utilizzando l'opzione di CPMD: `REAL SPACE WFN KEEP size`, che mantiene i valori delle trasformate di Fourier dirette in memoria (`size` va sostituito con la dimensione riservata in memoria dalle trasformate di Fourier dirette), si riesce a portare i tempi medi di esecuzione di un singolo passo di dinamica molecolare nei tre casi precedenti, rispettivamente a 9.8, 5.1 e 3.8 s come mostrato nella Tabella 4.

Da questi dati si vede come i tempi di esecuzione diminuiscano di alcuni secondi con miglioramento delle performance comprese tra il 10% e il 20%. Inoltre, l'altro aspetto che ci ha permesso di ridurre notevolmente i tempi di esecuzione delle simulazioni, e in particolare quelli relativi alla scrittura e lettura su file, è stata la possibilità di poter utilizzare il *filesystem* parallelo `GPFS` che è disponibile su CRESCO. Poiché la simulazione richiede la scrittura e lettura di file piuttosto grandi, come ad esempio i file di `RESTART` che raggiungono 1.2 GB, poter utilizzare `GPFS` al posto di `AFS`, permette la riduzione dei tempi di esecuzione di molte decine di minuti.

Per poter diminuire ulteriormente i tempi di esecuzione del codice, tenendo conto del fatto che nell'esecuzione parallela, il carico di lavoro viene suddiviso tra i vari processori in base al primo indice della *mesh* mediante la quale si calcolano le trasformate di Fourier 3D, abbiamo scambiato ciclicamente le coordinate del nostro sistema in modo tale che la *mesh* divenga 576x180x72. Questo ci ha permesso di abbassare il tempo medio di esecuzione per ogni passo di dinamica molecolare fino a 2 secondi quando il numero di core impiegati e pari a 576, ovvero al primo indice della *mesh*, come indicato nella Tabella 5.

Numero core	Time step medio	TASKGROUPS
72	9.78	1
144	5.09	2
216	3.85	3

Tabella 4. Tempi medi di esecuzione (s) per 300 passi di dinamica molecolare con `REAL SPACE WFN KEEP` size.

Numero core	Initialization time	Time step medio	Cpu time	Elapsed time
1	385.15	366.98	110491.27	113355.44
8	91.93	93.85	28253.82	29043.62
16	47.41	45.80	13795.74	14707.7
32	25.53	24.23	7302.14	8201.93
48	18.64	15.81	4772.17	6097.7
64	15.03	11.26	3418.29	8949.09
72	15.18	11.37	3438.83	4091.3
144	13.58	7.62	2317.9	3231.35
256	21.38	8.97	2740.42	3900.61
288	22.91	9.89	3006.15	3932.29
432	29.86	3.11	978.66	1926.53
576	63.00	2.06	700.54	1599.78
864	127.63	2.73	1017.67	1706.9

Tabella 5. Tempi medi di esecuzione (s) per 300 passi di dinamica molecolare con nuova *mesh*.

Attualmente, sempre allo scopo di ridurre i tempi di esecuzione delle simulazioni, si stanno effettuando ulteriori test per il nostro sistema in questa seconda configurazione, nei quali viene utilizzato sia il *filesystem* parallelo `GPFS` sia l'opzione di `CPMD REAL SPACE WFN KEEP size`.

4. Studio di una interfaccia idruro di magnesio (MgH_2) magnesio (Mg)

Per approfondire l'analisi dell'interfaccia Mg-MgH_2, dove si affacciano il piano (100) del magnesio a quello (010) dell'idruro di magnesio, e contemporaneamente testare le risorse di calcolo disponibili nei centri ENEA, abbiamo considerato un sistema costituito da un maggior numero di atomi (incremento del 50%) rispetto ai casi trattato nel rapporto RT-2017-12-ENEA. In particolare, il nuovo sistema è costituito 72 atomi di Mg da un lato e 60 "molecole" di MgH_2 dall'altro. In Figura 2c) viene mostrata la cella di simulazione del nostro sistema, dove gli atomi di magnesio sono rappresentati in due tonalità di azzurro a seconda che appartengano all'idruro di magnesio o al magnesio, mentre gli atomi di idrogeno sono rappresentati in colore verde.

Inizialmente abbiamo considerato separatamente i due blocchi di materiale, così come mostrato rispettivamente nella Figura 2a) e Figura 2b), e ne abbiamo calcolato la funzione d'onda che minimizza l'energia totale di ciascun sistema. In seguito, per ciascuna parte, è stato effettuato il rilassamento ionico per poi procedere allo studio del sistema completo.

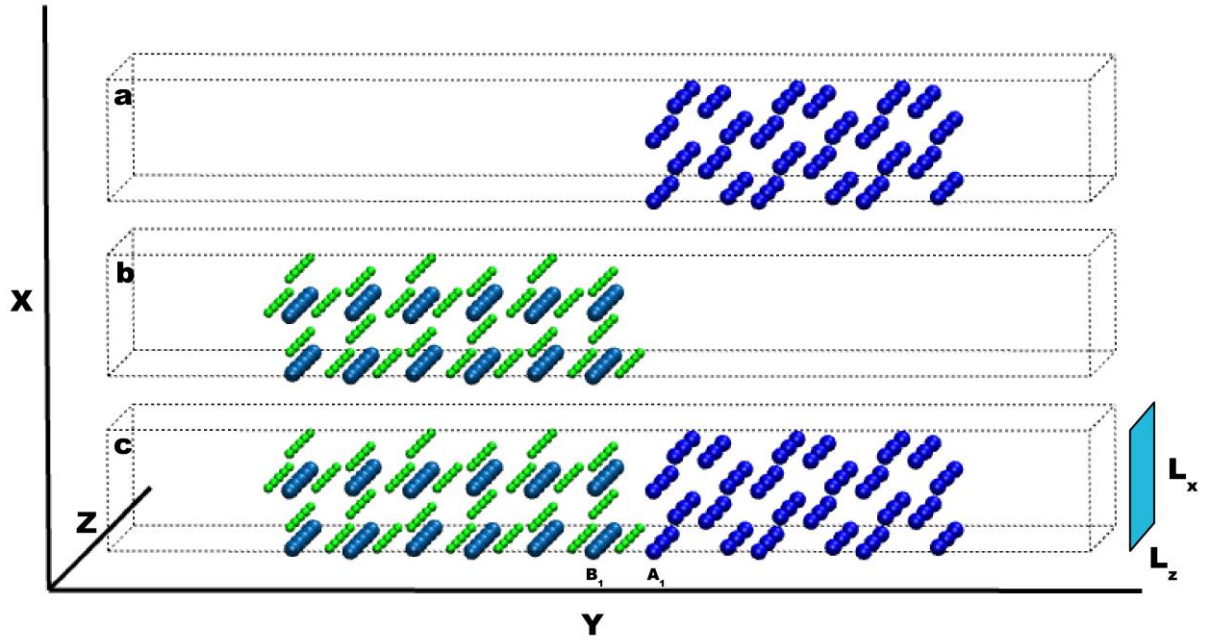

Figura 2. Celle di simulazione. a) lato magnesio; b) lato idruro di magnesio; c) interfaccia Mg-MgH_2.

4.1. Superficie libera di magnesio

Il sistema composto di magnesio (Figura 2a) è costituito da 72 atomi che sono disposti in modo tale da avere le due superfici esterne uguali, entrambe affacciano verso il vuoto ma una è libera mentre l'altra è mantenuta bloccata alle posizioni reticolari. In Figura 3 vengono mostrate le posizioni degli atomi di magnesio prima e dopo l'ottimizzazione geometrica, avendo mantenuto fissi gli ultimi due strati di atomi sulla destra. Ciò è stato fatto per simulare un sistema infinitamente esteso verso destra. Dalla figura si vede come a seguito del rilassamento ionico il sistema tenda a ristringersi, comportamento tipico per il magnesio e per altri metalli nel caso di superfici libere. Tra parentesi sono indicati gli spostamenti in Angstrom di ciascuna fila di atomi, da

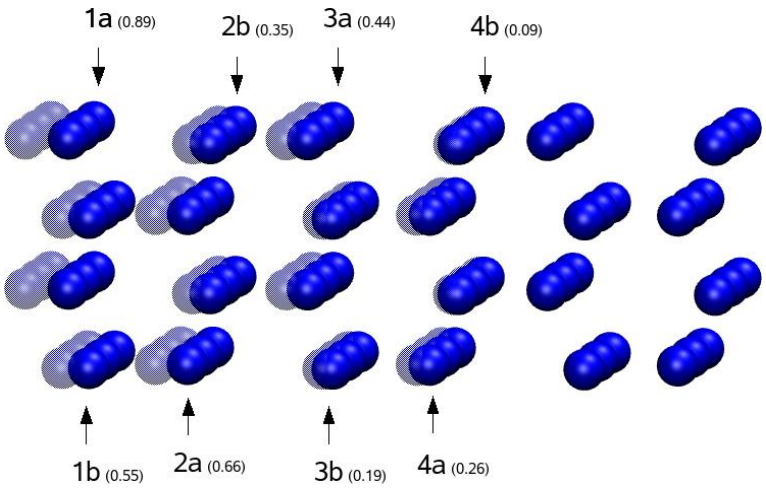

Figura 3. Rilassamento ionico della superficie libera di magnesio. Vengono indicati gli spostamenti (in Angstrom) per ciascuno strato di atomi.

tali spostamenti si vede come questi divengano sempre più piccoli allontanandosi dalla superficie libera.

4.2. Superficie libera di idruro di magnesio

La parte di sistema costituito dall'idruro di magnesio (Figura 2b) comprende 60 atomi di magnesio e 120 atomi di idrogeno. Questi si dispongono in 6 strati in modo tale da esporre verso l'esterno la stessa superficie; entrambe affacciano sul vuoto, ma una è libera mentre l'altra è mantenuta in posizione fissata con gli atomi bloccati alle loro posizioni reticolari. In Figura 4 vengono mostrate le posizioni degli atomi prima e dopo l'ottimizzazione geometrica avendo mantenuto fissi gli ultimi due strati di atomi di magnesio sulla sinistra. Ciò è stato fatto per simulare un sistema infinitamente esteso verso sinistra. Dalla Figura 4 si vede come a seguito del rilassamento ionico, a differenza del caso precedente, gli strati superiori tendano a spostarsi verso l'esterno aumentando lo spessore dello strato di materiale, mentre quelli inferiori tendano a muoversi verso l'interno. In figura, tra parentesi sono indicati gli spostamenti degli atomi di magnesio rispetto alle loro posizioni iniziali. Tali spostamenti sono minori rispetto a quelli del caso precedente; abbiamo indicato solo quelli relativi agli atomi più esterni perché gli altri atomi subiscono spostamenti molto piccoli. Va infine osservato che gli spostamenti degli atomi di idrogeno sono trascurabili.

4.3. Interfaccia idruro di magnesio - magnesio

Consideriamo ora il caso in cui questi due blocchi di materiale vengano affacciati uno di fronte all'altro come mostrato nella Figura 2c). Una volta trovata al distanza tra le superfici energeticamente più favorevole, abbiamo provveduto al rilassamento ionico mantenendo fissa la posizione degli atomi di magnesio più esterni. In analogia con i casi precedenti sono stati mantenuti fissi su ciascun lato, i due strati di atomi più esterni. Nella Figura 5 vengono mostrate due prospettive dell'interfaccia dopo l'ottimizzazione geometrica. Va osservato che in questo caso, a seguito del rilassamento ionico, dal lato del magnesio si verificano spostamenti che, anche se sono più piccoli, analogamente al caso della superficie libera, tendono a far diminuire lo spessore del materiale. Viceversa, dal lato dell'idruro, contrariamente al caso della superficie libera, si hanno

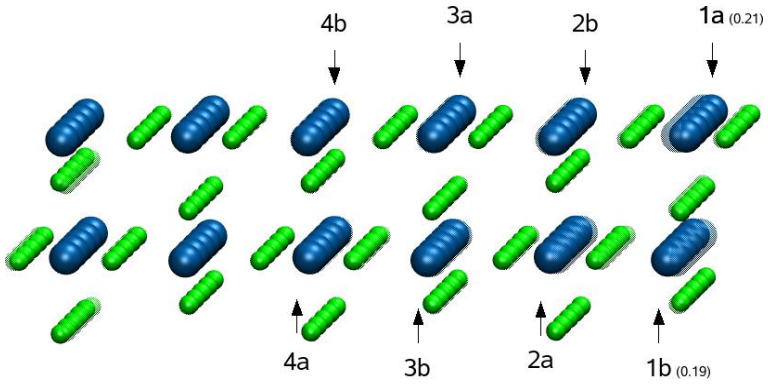

Figura 4. Rilassamento ionico della superficie libera di idruro di magnesio. Vengono indicati gli spostamenti (in Angstrom) per lo strato di atomi più esterno.

spostamenti in un'unica direzione che è quella in cui si ha un allargamento dello strato di materiale. Infine, va notato come ora gli atomi non si spostino più in maniera simmetrica, ma in qualche modo, risentendo ciascuna superficie dalla presenza dell'altra, gli atomi tendano ad posizionarsi in modo da far adattare tra di loro i profili delle due superfici (ciò è messo in evidenza nella Figura 4b). In modo del tutto analogo a quanto fatto nel rapporto RT-2017-12-ENEA, anche per questo sistema più grande, calcoliamo il lavoro di adesione, che risulta essere pari a 394.69 mJ/m^2.

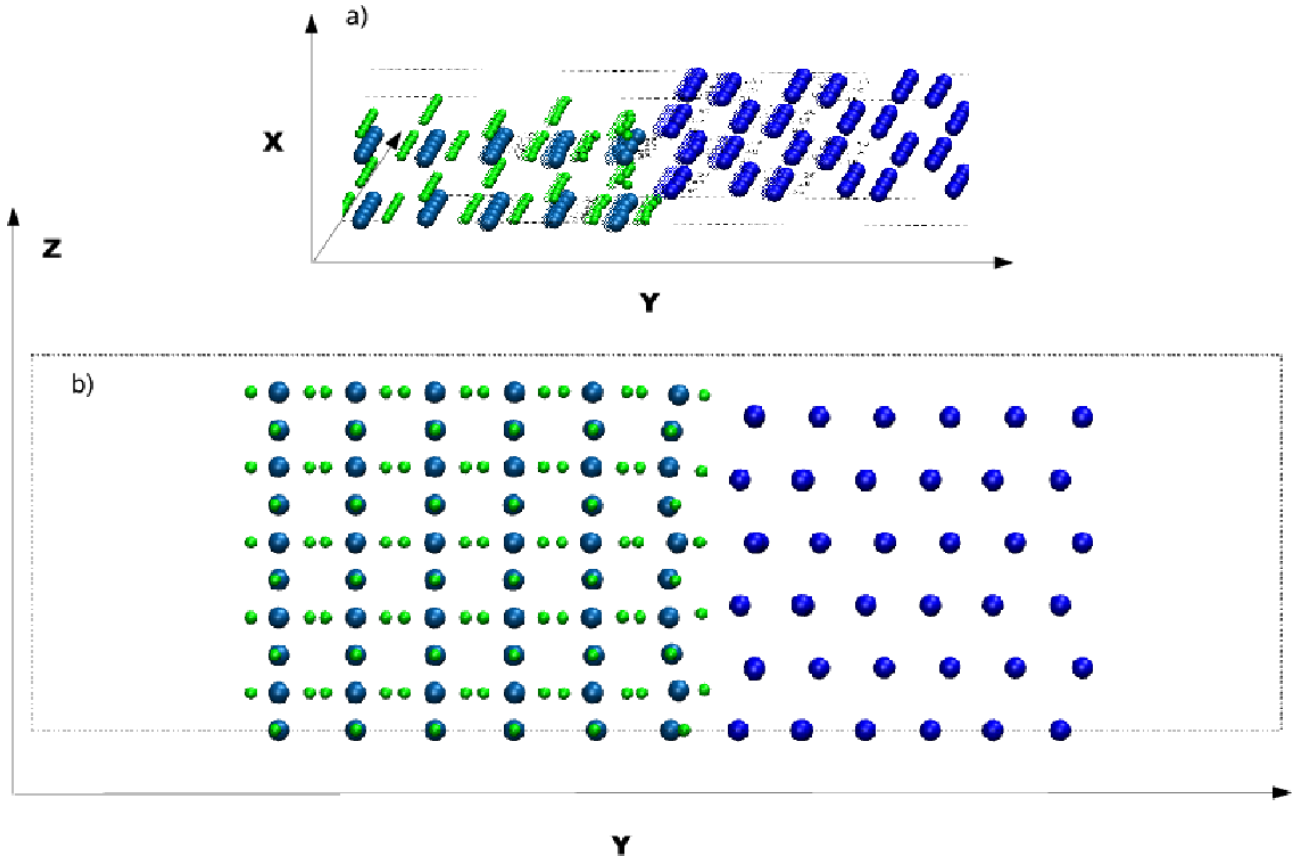

Figura 5. Due prospettive dell'interfaccia MgH_2 – Mg dopo il rilassamento ionico.

5. Studio della diffusione dell'idrogeno.

Dopo aver scelto le superfici di magnesio e idruro di magnesio che sono computazionalmente compatibili (devono essere rispettate le condizioni periodiche al contorno nelle direzioni x e z) per simulare un'interfaccia MgH_2 – Mg, e averne calcolato il lavoro di adesione, ora vogliamo studiare la mobilità degli atomi di idrogeno. In particolare siamo interessati a comprendere la dinamica a livello atomico, della diffusione dell'idrogeno sull'interfaccia, al variare della temperatura del sistema. Per far ciò abbiamo eseguito delle simulazioni di dinamica molecolare (MD) a volume e temperatura costanti partendo dalla temperatura ambiente (T = 300K) e arrivando a T= 900 K. In questo intervallo di temperatura, sperimentalmente si osserva la transizione di fase dall'idruro di magnesio (MgH_2**)** agli elementi separati (Mg + H_2), accompagnata dal desorbimento dell'idrogeno.

La Figura 6, mostrando quattro configurazioni atomiche a differenti temperature, mette in evidenza l'influenza della temperatura sulla mobilità degli atomi di idrogeno. Alla temperatura di T = 300 K (figura 6a) e T = 500 K (figura 6b) non c'è diffusione degli atomi di idrogeno, infatti, questa inizia a manifestarsi a T = 700 K (figura 6c). A questa temperatura c'è una chiara tendenza degli atomi di idrogeno a spostarsi verso l'interfaccia. La dinamica di questi spostamenti, che avvengono in prossimità dell'interfaccia, è caratterizzata da salti degli atomi di idrogeno tra posizioni reticolari vicine. Infine, a temperatura più alta, T = 900 K (figura 6d), gli atomi di idrogeno si muovono verso la superficie di magnesio abbandonando le loro posizioni reticolari. Questi risultati sono in buon accordo con quelli sperimentali, nei quali, nel caso in cui l'idruro di magnesio non sia stato sottoposto a processi di *milling* e non siano presenti catalizzatori, si inizia ad osservare il desorbimento dell'idrogeno nell'MgH_2 a T=780 K. A questa temperatura nelle simulazioni MD è chiaramente visibile la diffusione dell'idrogeno sull'interfaccia, sottolineando comunque che gli atomi di idrogeno non diffondono all'interno del reticolo cristallino del magnesio. Per quantificare in modo più preciso la diffusione degli atomi di idrogeno sull'interfaccia, in Figura 7 sono riportati i coefficienti di diffusione in funzione della temperatura. In questa figura viene evidenziato il diverso comportamento degli atomi di idrogeno in base alla loro collocazione spaziale. Infatti gli atomi più interni (*Bulk*) hanno una diffusività minore rispetto a quelli prossimi all'interfaccia (*Interface*), ed inoltre, tale differenza risulta più evidente alle alte temperature.

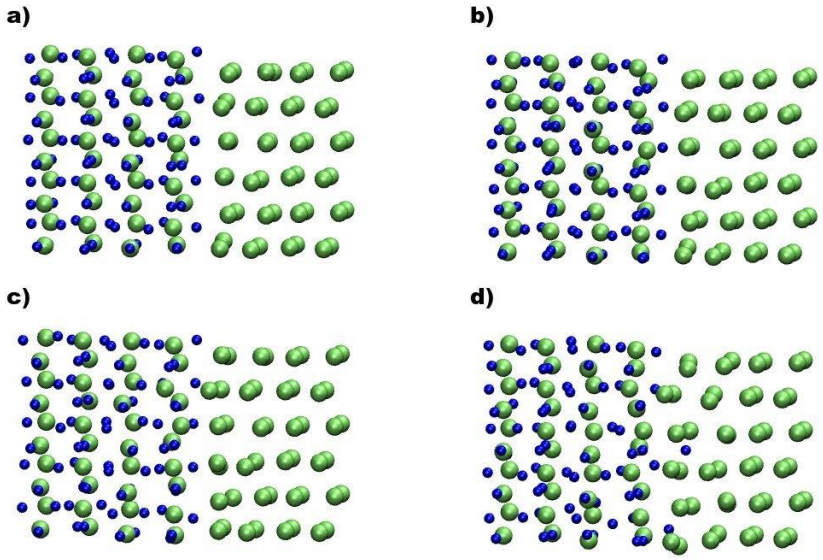

Figura 3. Snapshot dell'interfaccia MgH_2 - Mg a differenti temperature: a) T=300 K; b) T=500 K; c) T=700 K; d) T=900 K. Gli atomi di magnesio sono rappresentati in verde mentre quelli di idrogeno in blu**.**

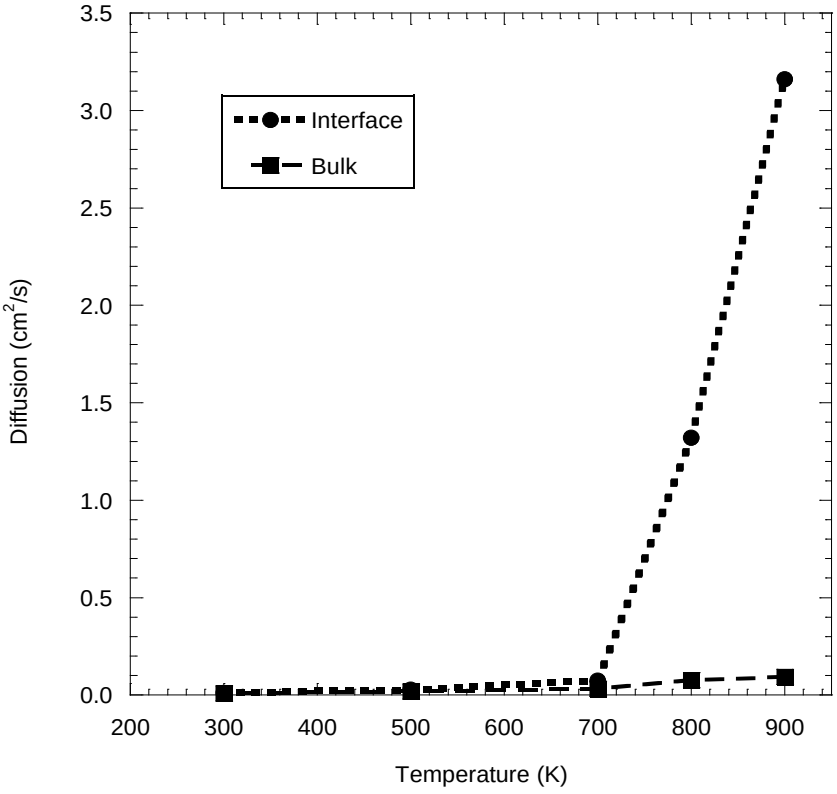

Figura 4. Coefficiente di diffusione degli atomi di idrogeno in funzione della temperatura.

ENEA
Servizio Promozione e Comunicazione
www.enea.it

Stampa: Laboratorio Tecnografico ENEA - C.R. Frascati
agosto 2017