MISSION INNOVATION ITALY

Ministero dell'Ambiente e della Sicurezza Energetica

D1.3 - Descrizione dell'utilizzo di CRESCO e della piattaforma IEMAP

Simone Giusepponi, Marco Catillo, Massimo Celino

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

Report MI21-24/130

D1.3 - DESCRIZIONE DELL'UTILIZZO DI CRESCO E DELLA PIATTAFORMA IEMAP

Simone Giusepponi (ENEA), Marco Catillo (ENEA), Massimo Celino (ENEA)

Dicembre 2024

Report MISSION INNOVATION

Ministero dell'Ambiente e della Sicurezza Energetica - ENEA
Mission Innovation 2021-2024 - III annualità

Progetto: Piattaforma accelerata per i Materiali per l'Energia
Work package: IEMAP: Italian Energy Materials Acceleration Platform
Linea di attività 1.2: Porting applicazioni su piattaforme HPC
Responsabile del Progetto: Massimo Celino, ENEA
Responsabile della LA: Filippo Palombi, ENEA

Indice

Sommario

Il presente documento riporta i dettagli relativi alle attività svolte nella LA1.3 i cui obiettivi definiti nel Piano di Attività sono l'assistenza agli utenti durante le sperimentazioni sulla piattaforma IEMAP, l'aggiornamento e manutenzione dei codici di calcolo e l'allocazione delle risorse destinate alle attività IEMAP mentre l'infrastruttura di supercalcolo CRESCO è in funzione e a disposizione di tutti gli utenti.

Le attività riportate in questo report sono necessarie e propedeutiche per l'aggiornamento nell'utilizzo della piattaforma di progettazione integrata e accelerata dei materiali per l'energia. In particolare, sono stati aggiornati a versioni più recenti quei codici che sono richiesti dagli utenti della piattaforma che necessitano dell'utilizzo dell'infrastruttura di calcolo ad alte prestazioni CRESCO e dei servizi ENEAGRID. Inoltre, a seguito della disponibilità di nuove risorse di calcolo, questi codici, necessari per lo studio di materiali innovativi per l'energia, sono stati compilati ed installati in ambiente ENEAGRID per loro uso efficace anche su questi due nuovi supercomputer: CRESCO7 e XCRESCO. Si riporta anche la descrizione e le modalità di utilizzo di un ambiente virtuale per l'accesso e la predisposizione delle librerie di machine learning delle quali si è provveduto al loro aggiornamento e al test su varie risorse di calcolo disponibili. Queste librerie sono necessarie per utilizzare tecniche di intelligenza artificiale per analizzare i dati disponibili nel database IEMAP.

1 Introduzione

L'attività infrastrutturale del progetto IEMAP è finalizzata alla predisposizione e alla manutenzione di tutti i servizi hardware/software necessari al corretto e completo sviluppo delle linee di attività proposte per l'automazione della ricerca di nuovi materiali anche attraverso tecniche di intelligenza artificiale. Come tale, essa costituisce un fondamento essenziale e propedeutico alla piena realizzazione delle finalità del progetto. Nella prima annualità sono stati creati gli account per l'accesso all'infrastruttura CRESCO/ENEAGRID, è stata svolta un'attività di formazione degli utenti riguardante le caratteristiche principali hardware e software, le modalità di accesso e l'utilizzo dell'infrastruttura CRESCO/ENEAGRID. Nella seconda annualità si è continuato su queste attività creando e formando nuove utenze e fornendo supporto agli utenti descrivendo in maggior dettaglio la piattaforma di supercalcolo CRESCO e tutti i servizi ad essa connessi all'interno di ENEAGRID.
Nell'ultima annualità si è continuato a garantire supporto agli utenti per garantire un uso efficiente dell'infrastruttura ENEAGRID. In particolare, poiché le risorse computazionali sono state potenziate come riportato nella Sezione 2 ed è stato arricchito l'ambiente di lavoro di ENEAGRID (Filesystem, gestore risorse e compilatori) come riportato nella Sezione 3, i codici di interesse per IEMAP sono stati aggiornati in modo da poter essere utilizzati in modo efficiente anche sui nuovi cluster HPC come indicato nella Sezione 4. Inoltre, nella Sezione 5 viene descritto come a questi software venga dedicata una area progettuale dedicata e, come il loro utilizzo sui vari cluster HPC sia facilitato dalla definizione di un *profile iemap* in ambiente *Environment Modules*. Infine, nella sezione 5, sono descritte ampliamente le attività di implementazione e utilizzo di librerie di machine learning. Queste librerie sono state ulteriormente sviluppate e aggiornate descrivendo anche in modo dettagliato come poterle utilizzare al meglio. Con questa finalità si sono valutate le performance di varie risorse di calcolo disponibili in ENEAGRID.

2 Risorse di calcolo disponibili in ENEAGRID

In questa sezione verranno elencati e descritte le risorse computazionali HPC (High-Performance Computing) disponibili in ENEAGRID anche a seguito dell'aggiunta di un cluster (CRESCO7) presso il centro ricerche ENEA di Portici e di uno (XCRESCO) presso il centro ricerche ENEA di Frascati.

2.1 Centro ricerche ENEA Portici

Presso il centro di ricerche ENEA di Portici sono disponibili i seguenti cluster HPC:

- **CRESCO6**: sistema di calcolo basato su architettura x86_64 con potenza di picco pari a 1.4 PetaFlops, costituito da 434 nodi interconnessi tra loro da una rete a larga banda e bassa latenza basata su Intel Omni-Path a 100 Gb/s. Ogni nodo è equipaggiato con due socket con processore a 24 core Intel Xeon Platinum 8160 con frequenza di clock pari 2.10 GHz e 192 GB di RAM, disco da 500 GB SATA II, interfaccia Intel Omni-Path 100 Gb/s, due interfacce GbE e supporto BMC/IPMI 1.8 con software per la gestione remota della console. Si hanno quindi a disposizione complessivamente 20.832 core. Il sistema operativo installato è CentOS Linux release 7.3 e i file system disponibili sono: AFS che è il file system geograficamente distribuito comune a tutta ENEAGRID e GPFS che è il file system di IBM ad alte prestazioni per l'I/O parallelo. Per la sottomissione di lavori computazionali a questa infrastruttura e la conseguente gestione delle risorse il batch scheduler utilizzato è LSF Spectrum di IBM.

- **CRESCO7**: sistema di calcolo basato su architettura x86_64 con potenza di picco pari a 500 TeraFlops, costituito da 144 nodi interconnessi tra loro da una rete a larga banda e bassa latenza basata su Mellanox-EDR a 100 Gb/s. Ogni nodo è equipaggiato con due socket con processore a 24 core Intel Xeon Platinum 8160 con frequenza di clock pari 2.10 GHz e 192 GB di RAM, disco da 500 GB SATA II, interfaccia Mellanox-EDR a 100 Gb/s, due interfacce GbE e supporto BMC/IPMI 1.8 con software per la gestione remota della console. Si hanno quindi a disposizione complessivamente 6912 core. Il sistema operativo installato è AlmaLinux release 9.2 e i file system disponibili sono: AFS che è il file system geograficamente distribuito comune a tutta ENEAGRID e LUSTRE che è un file system ad alte prestazioni per l'I/O parallelo. Per la sottomissione di lavori computazionali a questa infrastruttura e la conseguente gestione delle risorse il batch scheduler utilizzato è SLURM.

2.2 Centro ricerche ENEA Frascati

Presso il centro di ricerche ENEA di Frascati sono disponibili i seguenti cluster HPC:

- **CRESCO4F**: sistema di calcolo basato su architettura x86_64 con potenza di picco pari a 20 TeraFlops, costituito da 64 nodi interconnessi tra loro da una rete a larga banda e bassa latenza basata su Infiniband 4xQDR a 40 Gb/s. Ogni nodo è equipaggiato con due socket con processore a 8 core Intel E5-2670 con frequenza di clock pari 2.60 GHz e 64 GB di RAM, disco da 500 GB SATA II, interfaccia Infiniband QDR a 40 Gb/s, due interfacce GbE e supporto BMC/IPMI 1.8 con software per la gestione remota della console. Si hanno quindi a disposizione complessivamente 1024 core. Il sistema operativo installato è CentOS Linux release 7.8 e i file system disponibili sono: AFS che è il file system geograficamente distribuito comune a tutta ENEAGRID e GPFS che è il file system di IBM ad alte prestazioni per l'I/O parallelo. Per la sottomissione di lavori computazionali a questa infrastruttura e la conseguente gestione delle risorse il batch scheduler utilizzato è LSF Spectrum di IBM.
- **XCRESCO**: sistema di calcolo basato su architettura ppc64le con potenza di picco pari a 1.9 PetaFlops, costituito da 60 nodi interconnessi tra loro da una rete a larga banda e bassa latenza basata su Mellanox-EDR a 100 Gb/s. Ogni nodo è equipaggiato con due socket con processore a 16 core (hyperthreading x4) IBM Power 9 AC922 con frequenza di clock pari 2.6(3.1) GHz e 256 GB di RAM e interfaccia Mellanox-EDR a 100 Gb/s. Inoltre, sono disponibili come acceleratori GPU (Graphics Processing Unit), 4 schede NVIDIA Tesla V100 con 16 GB di memoria con supporto Nvlink 2.0 per l'interconnessione a coppie tra le GPU. Si hanno quindi a disposizione complessivamente 1920 core fisici (7680 core logici) e 240 GPU. Il sistema operativo installato è AlmaLinux release 9.2 e i file system disponibili sono: AFS che è il file system geograficamente distribuito comune a tutta ENEAGRID e NFS che è un file system in grado di gestire dispositivi di memorizzazione remoti. Per la sottomissione di lavori computazionali a questa infrastruttura e la conseguente gestione delle risorse il batch scheduler utilizzato è SLURM.

2.3 Centro ricerche ENEA Casaccia

Presso il centro di ricerche ENEA di Casaccia (Roma) è disponibile il seguente cluster HPC:

- **CRESCO4C**: sistema di calcolo basato su architettura x86_64 con potenza di picco pari a 10 TeraFlops, costituito da 32 nodi interconnessi tra loro da una rete a larga banda e bassa latenza basata su Infiniband 4xQDR a 40 Gb/s. Ogni nodo è equipaggiato con due socket con processore a 8 core Intel E5-2670 con frequenza di clock pari 2.60 GHz e 64 GB di RAM, disco da 500 GB SATA II, interfaccia

Infiniband QDR a 40 Gb/s, due interfacce GbE e supporto BMC/IPMI 1.8 con software per la gestione remota della console. Si hanno quindi a disposizione complessivamente 512 core. Il sistema operativo installato è CentOS Linux release 7.8 e i file system disponibili sono: AFS che è il file system geograficamente distribuito comune a tutta ENEAGRID e GPFS che è il file system di IBM ad alte prestazioni per l'I/O parallelo. Per la sottomissione di lavori computazionali a questa infrastruttura e la conseguente gestione delle risorse il batch scheduler utilizzato è LSF Spectrum di IBM.

La disponibilità di questi due nuovi cluster HPC (CRESCO7 e XCRESCO) ha aumentato le capacità di calcolo dell'intera infrastruttura e ha introdotto nuovi elementi hardware e software relativi all'architettura (x86_64 e ppc64le) agli acceleratori (GPU), tipologie di filesystem (LUSTRE e NFS), gestione delle risorse (SLURM) e sistema operativo (AlmaLinux) con relative suite di compilatori e librerie matematiche. Questi elementi saranno brevemente descritti nella sezione seguente. Infine, all'interno dell'infrastruttura, sono disponibili ulteriori nodi con GPU e/o grande RAM dedicati ad attività specifiche quali la grafica avanzata e gestione di grandi quantità di dati, come ad esempio le risorse CRESCO5F e CRESCOB che sono state utilizzate come descritto nella sezione 6.

3 Filesystem, gestore risorse e compilatori disponibili in ENEAGRID

In questa sezione presentiamo l'ambiente di lavoro base di ENEAGRID necessario per utilizzare l'infrastruttura HPC in modo efficiente, che comprende i filesystem, i gestori delle risorse e le suite di compilatori con librerie numeriche associate.

3.1 Filesystem

In ENEAGRID sono utilizzati differenti tipi di filesystem con la finalità di soddisfare, tra le altre, due esigenze distinte. Da una parte, la necessità di avere un filesystem condiviso tra i vari cluster HPC e dall'altra, avere filesystem con elevate performance di lettura e scrittura dei dati che vengono prodotti dalle simulazioni, in particolare, da quei codici che fanno un ampio utilizzo di I/O parallelo dove numerosi processi accedono agli stessi dati in contemporanea. Alla prima esigenza risponde il filesystem OpenAFS, alla seconda i filesystem GPFS, LUSTRE e NFS.

- **OpenAFS** è l'implementazione open-source di AFS (Andrew File System) che è un filesystem geograficamente distribuito che permette di avere una visione "locale" e "uniforme" dei file dislocati su diversi nodi della griglia. Ogni utente di ENEAGRID possiede la propria HOME in AFS, in modo da poter accedere ai propri file e a tutte le risorse da qualsiasi postazione e indipendentemente dal cluster sul quale accede. La gestione dei permessi per l'accesso ai dati è garantita mediante l'utilizzo di apposite ACL (Access Control List).
- **GPFS** (General Parallel File System) **Spectrum Scale** è un filesystem ad alte prestazioni ottimizzato per accesso parallelo ai dati sviluppato da IBM. GPFS garantisce, alle applicazioni che lo utilizzano, accesso concorrente a bassa latenza all'interno di un cluster mediante interconnessione veloce a bassa latenza Infiniband o Omnipath in base al cluster. Questo filesystem è disponibile su CRESCO6, CRESCO4F e CRESCO4C.
- **LUSTRE** è un filesystem open-source ad alte prestazioni ottimizzato per accesso parallelo ai dati. Uno dei principali fattori che determinano le elevate prestazioni del filesystem LUSTRE è la capacità di eseguire la stripe dei dati su più target di archiviazione (OST) in modo round-robin. I file possono

essere suddivisi in più parti che verranno poi memorizzate su diversi OST all'interno del sistema LUSTRE. Questo filesystem è disponibile su CRESCO7.

- **NFS** (Network File System) è un filesystem che consente a un cluster HPC con numerosi nodi di calcolo (client) di utilizzare la rete per accedere a file e directory condivise da server remoti come fossero disponibili in locale. Questo filesystem è disponibile su XCRESCO.

3.2 Gestore delle risorse

Il gestore delle risorse è una applicazione software che permette di gestire le risorse di calcolo dei cluster HPC. Questo è necessario in quanto ad uno stesso cluster HPC accedono numerosi utenti che fanno richiesta di risorse di calcolo per eseguire i propri lavori computazionali (job). Perciò, la finalità del batch scheduler è quella di gestire l'infrastruttura HPC soddisfacendo alle richieste computazionali dei singoli job ma evitando che ci siano job concorrenti sulla stessa risorsa. Questo avviene mediante un accodamento dei job che vengono eseguiti non appena sono disponibili le risorse computazionali richieste. In ENEAGRID sono utilizzati due differenti gestori delle risorse.

- **IBM Spectrum LSF** (Load Sharing Facility) è una suite di software IBM per la distribuzione del carico di lavoro in sistemi distribuiti. LSF fornisce un framework di gestione delle risorse che prende i requisiti del job, trova le risorse migliori per eseguire il job e ne monitora l'avanzamento. Questo software è utilizzato per la gestione delle risorse dei cluster CRESCO6, CRESCO4F e CRESCO4C.
- **SLURM Workload Manager** (Simple Linux Utility for Resource Management) è un sistema open-source, tollerante ai guasti e altamente scalabile per la gestione dei cluster e la schedulazione dei job in cluster Linux di grandi e piccole dimensioni. Questo software è utilizzato per la gestione delle risorse dei cluster CRESCO7 e XCRESCO.

3.3 Suite compilatori

In questa sezione verranno menzionate le varie suite di compilatori disponibili sui vari cluster e le implementazioni di librerie MPI (Message Passing Interface) utilizzabili. I compilatori permettono la compilazione dei file sorgente (scritti in vari linguaggi di programmazione come C, C++, FORTRAN ecc.) di un software, per la creazione di file eseguibili da poter essere utilizzati sul cluster scelto. Invece, MPI è un modello di programmazione e una libreria di standard per la comunicazione tra processi paralleli nel calcolo ad alte prestazioni e applicazioni distribuite. Sono disponibili diverse implementazioni di MPI: openmpi, impi e mpich2. Poiché l'intera infrastruttura HPC è costituita da vari cluster che hanno architetture differenti (x86_64 e PowerPC) con sistemi operativi differenti (CentOS 7.3, CentOS 7.8 e AmaLinux 9.2) ed è utilizzata da centinaia di utenti che la usano per varie applicazioni in molti campi di ricerca (energia, ambiente, nucleare, informatica, biotech ecc.), si hanno a disposizione vare versioni di suite di compilatori così come implementazioni di MPI per venire incontro all'utenza che utilizza codici di versioni più o meno recenti scritti con vari linguaggi di programmazione. Di seguito riportiamo le tre principali suite di compilatori con le implementazioni di MPI connesse disponibili sui tre cluster HPC principali (CRESCO6, CRESCO7 e XCRESCO)

- **GNU** Compiler Collection (GCC) è una suite di compilatori ottimizzante multipiattaforma come parte del progetto GNU. Disponibile per vari linguaggi, architetture e sistemi operativi, è distribuito in forma libera dalla Free Software Foundation. Su CRESCO6 è disponibile la versione 7.3.0 con openmpi 3.1.2 e la versione 9.3.0 con openmpi 4.0.5. Su CRESCO7 sono disponibili le versioni 7.3.0, 9.3.0 e 11.3.1, con la versione 4.1.6 per openmpi associata a GCC 11.3.1. Infine, su XCRESCO sono

disponibili le versioni 6.4.0, 8.3.0 e 11.3.1. Associate a GCC 11.3.1 ci sono le versioni 4.1.1 e 4.1.6 per openmpi, dove quest'ultima essendo "cuda aware" permettendo lo scambio diretto di dati tra GPU.

- **Intel** fornisce una sua suite di compilatori per i vari linguaggi di programmazione che produce eseguibili particolarmente performanti su cluster di nodi con processori Intel anche grazie alla presenza delle librerie matematiche ottimizzate MKL (Math Kernel Library). Su CRESCO6 sono disponibili le versioni intel16, intel17, intel18 e intel19 con le impi associate. Inoltre, ci sono le openmpi 3.1.2 per intel17 e intel18 e le openmpi 4.0.3 per intel19. Su CRESCO7 sono disponibili le versioni intel17, intel18, intel/oneapi2023 e intel/oneapi2024 con le impi associate. Inoltre, è disponibile la versione 4.1.5 di openmpi per intel/oneapi2023.
- **NVHPC** è la suite di compilatori fornita da NVIDIA che supporta gli acceleratori GPU. Inoltre, fornisce librerie matematiche ottimizzate per GPU e librerie di comunicazione ottimizzate per la comunicazione tra varie GPU. Questa suite è disponibile solo sul cluster XRESCO in quanto è l'unico equipaggiato con acceleratori GPU con versione di CUDA 12.3. La versione disponibile è la 24.1 con versione 4.1.5 di openmpi.

4 Codici di interesse per IEMAP

In questa sezione verranno elencati i codici di simulazione ad alto parallelismo resi disponibili e utilizzati nell'ambito del progetto IEMAP evidenziando gli aggiornamenti alle versioni più recenti e la loro disponibilità sui vari cluster HPC disponibili. I codici descritti sono Quantum ESPRESSO, CP2K e VASP, che adottano approcci quanto-meccanici e LAMMPS che invece adotta tecniche di simulazione di dinamica classica. Codici che richiedendo elevate capacità di calcolo sono stati compilati e installati per essere utilizzati sui tre cluster HPC con potenza di calcolo maggiore, ovvero, CRESCO6, CRESCO7 e XCRESCO. Si è cercando di avere versioni dei codici recenti e uguali sui tre cluster ma allo stesso tempo compatibili con la dotazione dei compilatori disponibili.

4.1 Quantum ESPRESSO

Quantum ESPRESSO (QE) è una suite integrata di codici numerici open-source per il calcolo della struttura elettronica e il modeling dei materiali a scala nanometrica. I codici si basano sulla teoria del funzionale densità (DFT - Density Functional Theory) ed utilizzano la base delle onde piane e gli pseudopotenziali (www.quantum-espresso.org) (Giannozzi, 2009).
Nel corso degli anni Quantum ESPRESSO si è evoluto in una distribuzione di codici indipendenti ma interoperabili tra di loro mantenendo lo spirito di un progetto open-source (rilasciato sotto licenza GNU-GPL). Questa distribuzione fornisce un insieme di codici "storici" e un insieme di codici "plug-in" che effettuano compiti specifici più avanzati, ed in aggiunta, un numero di pacchetti da terze parti concepiti in modo da essere interoperabili con i codici fondamentali. La distribuzione completa contiene i seguenti pacchetti per il calcolo delle proprietà di struttura elettronica basate sulla teoria funzionale densità con l'utilizzo della base ad onde piane e pseudopotenziali: PWscf, CP, PWneb, PHonon, PostProc, PWcond, GWL, XSPECTRA, TDDFPT, EPW. La suite Quantum ESPRESSO è utilizzabile su diversi tipi di architettura che vanno dalle più potenti infrastrutture di calcolo parallelo alle workstation fino ai personal computer e permette l'esecuzione dei seguenti tipi di calcoli:

- Calcoli di stato fondamentale;
- Ottimizzazione strutturale, dinamica molecolare, superfici di energia potenziale;

- Elettrochimica e condizioni al contorno speciali;
- Proprietà di risposta (teoria perturbativa del funzionale densità);
- Proprietà spettroscopiche;
- Trasporto quantistico.

La versione di Quantum ESPRESSO è stata aggiornata alla versione 7.2 che è disponibile su tutti e tre i cluster. Su CRESCO6 è stata compilata nella versione seriale e parallela con intel19 e impi/intel19. Su CRESCO7 è stata compilata nella versione seriale e parallela con intel/oneapi2023 e impi/oneapi2023. Su XCRESCO è stata compilata la versione parallela che utilizza gli acceleratori GPU utilizzando la suite nvhpc 24.1 con CUDA 12.3 e openmpi 4.1.5.

4.2 CP2K

CP2K è un codice numerico di chimica computazionale e fisica dello stato solido che permette simulazioni atomistiche di sistemi solidi, liquidi, molecolari e biologici. CP2K fornisce una struttura generale per differenti metodi di modeling come, ad esempio, la teoria del funzionale densità con l'utilizzo misto di Gaussiane e onde piane (GPW) e pseudopotenziali, ma è anche possibile fare calcoli all-elettron, oppure, utilizzare come basi solo onde piane o solo Gaussiane (www.cp2k.org) (Kühne, 2020).
I livelli teorici supportati includono DFTB, LDA, GGA, MP2, RPA, metodi semi empirici (ad esempio AM1, PM3, PM6, RM1, MNDO) e force fields classici (ad esempio AMBER, CHARMM). Con CP2K si possono effettuare simulazioni di dinamica molecolare (non quella Car-Parrinello), meta-dinamica, Monte Carlo, dinamica Ehrenfest, analisi vibrazionali, minimizzazioni dell'energia e ottimizzazione di transizione di stato usando NEB o dimer method. CP2K è scritto in Fortran 2008 ed è rilasciato gratuitamente mediante una licenza GPL. Può girare efficientemente in parallelo potendo usare una combinazione di strategie di parallelizzazione (OpenMP, MPI e CUDA) con scalabilità lineare permettendo i seguenti tipi di calcoli:

- Metodi di teoria della struttura elettronica ab-initio usando il modulo QUICKSTEP;
- Dinamica Molecolare ab-initio;
- Simulazioni miste classico-quantistiche (QM/MM).

La versione di CP2K è stata aggiornata alla versione 2023.2 e questa versione è disponibile su CRESCO6 e CRESCO7. Su CRESCO6 è stata compilata la versione parallela con GCC 9.3.0 e openmpi 4.0.5. Su CRESCO7 è stata compilata la versione parallela con GCC 11.3.1 e openmpi 4.1.6.

4.3 VASP

Il codice VASP (Vienna Ab-initio Simulation Package) è un software per il modeling di materiali a scala atomica, cioè calcoli di struttura elettronica e dinamica molecolare quanto-meccanica ab-initio (www.vasp.at) (Hafner, 2007).
VASP calcola una soluzione approssimata all'equazione di Schrödinger multi-corpo, sia mediante la teoria del funzionale densità (DFT) risolvendo le equazioni di Kohn-Sham, sia utilizzando l'approssimazione di Hartree-Fock mediante la risoluzione delle equazioni di Roothaan. Sono inoltre implementati i funzionali ibridi che combinano l'approccio di Hartree-Fock con la teoria del funzionale densità. In aggiunta, VASP rende disponibili i metodi delle funzioni di Green (GW e ACFDT-RPA) e la teoria della perturbazione a multi-corpo (Møller-Plesset al secondo ordine). In VASP, le quantità fondamentali come gli orbitali elettronici, la densità di carica elettronica e il potenziale locale, sono espressi mediante la base delle onde piane. Le interazioni tra

gli elettroni e gli ioni sono descritte usando pseudopotenziali a norma conservata o ultrasoft, oppure attraverso il metodo projector-augmented-wave (PAW). Per determinare lo stato elettronico fondamentale, VASP fa uso di efficienti tecniche iterative di diagonalizzazione delle matrici, come il metodo di minimizzazione residua con inversione diretta dello spazio iterativo (RMM-DIIS) o algoritmi di Davidson a blocchi. Questi sono accoppiati con schemi di Broyden e Pulay altamente efficienti per accelerare i cicli di auto-consistenza. Il codice VASP ha, tra le altre, implementate le seguenti funzionalità:

- Rilassamento strutturale;
- Funzionali;
- Dinamica molecolare;
- Metodi della funzione di Green;
- Teoria della perturbazione multi-corpo;
- Risposta lineare ai campi elettrici e agli spostamenti ionici;
- Proprietà ottiche.

VASP è un codice licenziato a pagamento e la versione installata in ENEAGRID è la 6.2.1. Questa versione è stata compilata su CRESCO6 con intel19 e impi/intel19 e su CRESCO7 con intel/oneapi2023 e impi/oneapi2023. L'utilizzo di questo software è limitato agli utenti con licenza, mediante la definizione delle ACL (Acces Control List) di AFS.

4.4 LAMMPS

LAMMPS (Large-scale Atomic/Molecular Massively Parallel Simulator) è un codice di dinamica molecolare classica che simula insiemi di particelle allo stato solido, liquido e gassoso. Può modellare sistemi atomici, polimerici, biologici, a stato solido (metalli, materiali ceramici, ossidi), granulari, coarse-grained, o sistemi macroscopici usando una varietà di potenziali interatomici (campi di forza) e condizioni al contorno. Può modellare sistemi bi-dimensionali o tri-dimensionali con dimensioni che possono andare da alcune a miliardi di particelle (www.lammps.org) (Thompson, 2022).
Questo codice è scritto in C++ e può essere compilato e utilizzato su singolo PC, anche se è stato progettato per computer paralleli che supportano MPI. Questi includono workstation multicore a memoria condivisa, server/cluster multi-CPU a memoria distribuita e supercomputer. Inoltre, parti di LAMMPS supportano parallelizzazione multithreading OpenMP e accelerazione con GPU. LAMMPS è un codice open-source liberamente disponibile, distribuito sotto i termini della licenza GNU, che lo rende usabile e modificabile a piacimento. LAMMPS integra le equazioni del moto di Newton per un insieme di particelle interagenti. Per particella si può considerare un atomo, una molecola, un elettrone, oppure un cluster di atomi o blocchi di materiale mesoscopico e macroscopico. I modelli di interazione che LAMMPS include sono principalmente quelli a corto raggio, anche se, sono disponibili alcuni a lungo raggio.
Su macchine parallele vengono utilizzate tecniche di decomposizione spaziale che, mediante la parallelizzazione MPI, partiziona il dominio di simulazione in sottodomini di ugual costo computazionale assegnandone uno a ciascun processore/core. I processori/core comunicano e memorizzano le informazioni degli atomi "ghost" che circondano il proprio sottodominio.
La versione di LAMMPS è stata aggiornata alla versione del 29/08/2024 che è disponibile su tutti e tre i cluster. Su CRESCO6 è stata compilata nella versione seriale e parallela con GCC 9.3.0 e openmpi 4.0.5. Su CRESCO7 è stata compilata nella versione seriale e parallela con intel/oneapi2023 e openmpi 4.1.5. Su XCRESCO è stata compilata la versione parallela che utilizza gli acceleratori GPU utilizzando la suite nvhpc

24.1 con CUDA 12.3 e openmpi 4.1.5. LAMMPS è organizzato in "pacchetti", ciascuno con le proprie funzionalità che possono essere installati indipendentemente uno dall'altro. La versione disponibile ha inclusi i seguenti pacchetti: MOLECULE KSPACE AMOEBA ASPHERE BOCS BODY BPM BROWNIAN CG-DNA CG-SPICA CLASS2 COLLOID CORESHELL DIFFRACTION DIPOLE DPD-BASIC DPD-MESO DPD-REACT DPD-SMOOTH DRUDE EFF EXTRA-COMMAND EXTRA-COMPUTE EXTRA-DUMP EXTRA-FIX EXTRA-MOLECULE EXTRA-PAIR FEP GRANULAR INTERLAYER MANYBODY MC MEAM MESONT MISC ML-SNAP MOFFF ORIENT PERI PLUGIN QEQ REACTION REAXFF REPLICA RIGID SHOCK SPH SPIN SRD UEF YAFF DIELECTRIC ML-IAP PHONON. Il pacchetto GPU è disponibile solo per la versione utilizzabile sul cluster XCRESCO che equipaggiato con acceleratori GPU.

5 Area progettuale dedicata e allocazione risorse computazionali

Con la finalità di destinare i codici sopra citati ai soli utenti del progetto IEMAP, è stata realizzata un'area progettuale dedicata a questo scopo nel filesystem AFS al path /afs/enea.it/project/iemap_software/. Questo permette di rendere visibile quest'area a tutti i cluster HPC. Inoltre, mediante l'utilizzo delle ACL di AFS, l'accesso a questa area è permesso ai soli utenti del progetto IEMAP per i quali è stato definito un gruppo specifico. Per rendere più agevole ed efficiente l'infrastruttura di supercalcolo, è stata utilizzata una funzionalità di *Environment Modules*. Il sistema Environment Modules è uno strumento software che aiuta gli utenti a gestire il proprio ambiente shell Unix o Linux, consentendo di creare o rimuovere dinamicamente gruppi di impostazioni di variabili di ambiente correlate in modo tale da caricare ambienti diversi a seconda delle applicazioni o librerie necessarie. A questa finalità è stato definito un *profile iemap* che, una volta caricato, rende disponibili in automatico i moduli associati ai codici dedicati al progetto IEMAP sul cluster che si vuole utilizzare. Quindi, indipendentemente dal cluster su cui l'utente accede, dando il comando (eseguibile dai soli utenti del progetto):

```
module load /afs/enea.it/project/iemap_software/profile/iemap
```

viene caricato il *profile iemap* che rende visibili i seguenti moduli:

- su CRESCO6

```
--------------------------- /afs/enea.it/project/iemap_software/modules/cresco6 ---------------------------
cp2k/2023.2      lammps/lmp24_par lammps/lmp24_ser qe/7.2_par       qe/7.2_ser       vasp/6.2.1_par
<giuseps@cresco6x001 ~>
```

- su CRESCO7

```
--------------------------- /afs/enea.it/project/iemap_software/modules/cresco7 ---------------------------
cp2k/2023.2  lammps/lmp24_par  lammps/lmp24_ser  qe/7.2_par  qe/7.2_ser  vasp/6.2.1_par

Key:
loaded  default-version  modulepath
<giuseps@cresco7x001 ~>
```

- su XCRESCO

```
--------------------------- /afs/enea.it/project/iemap_software/modules/xcresco ----------------------------
lammps/lmp24_gpu  qe/7.2_gpu

Key:
loaded  modulepath  default-version
<giuseps@xcrescox001 ~> 
```

Questi moduli sono associati alla versione del codice appropriata al cluster specifico. Inoltre, agli utenti sono state fornite le indicazioni per la sottomissione di *job* sui cluster insieme a degli *script* di lancio di riferimento. Ad esempio, se si vuole utilizzare la versione parallela del codice pw.x della suite di Quantum ESPRESSO su CRESCO7, si deve caricare il modulo desiderato con il comando

```
module load qe/7.2_par
```

e poi utilizzare uno *script.sh* di sottomissione di questo tipo

```
#!/usr/bin/pagsh

#SBATCH --job-name=nome_job
#SBATCH --err=%j.err
#SBATCH --out=%j.out
#SBATCH --time=00:10:00
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=48
#SBATCH --ntasks-per-socket=24
#SBATCH --cpus-per-task=1
#SBATCH --partition=cresco7
#SBATCH --account=iemap
#SBATCH --auks=yes

aklog

mpirun pw.x -i file_input.in > file_output.out  + args
```

nel quale definisco: il nome del job, i nomi dei file su cui scrivere lo std_err e lo std_out; richiedo: 4 nodi (48 tasks per node), la risorsa cresco7 e l'account dedicato IEMAP. Nell'ultima riga sono specificati il nome del codice (pw.x), il file di input e il file di output e gli eventuali argomenti da fornire al codice.
Analogamente, se si vuole utilizzare la versione parallela di LAMMPS su XCRESCO, si deve caricare il modulo desiderato con il comando

```
module load lammps/lmp24_gpu
```

e poi utilizzare uno *script.sh* di sottomissione di questo tipo

```
#!/usr/bin/pagsh

#SBATCH --job-name=nome_job
```

```
#SBATCH --err=%j.err
#SBATCH --out=%j.out
#SBATCH --time=00:10:00
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=32
#SBATCH --ntasks-per-socket=16
#SBATCH --cpus-per-task=1
#SBATCH –gres=gpu:4
#SBATCH --partition=xcresco
#SBATCH --account=iemap
#SBATCH --auks=yes

aklog

mpirun lmp24_gpu -in file_input.in -log file_log.log  + args
```

dove, rispetto a prima c'è la voce che permette di utilizzare le GPU (–gres=gpu:4). In entrambi i casi la sottomissione avviene dando il comando

```
sbatch script.sh
```

che dà a SLURM la gestione del job. Il controllo del job, come ad esempio, vedere lo stato di avanzamento ecc. avviene mediante i comandi specifici di SLURM. Infine, per la sottomissione su CRESCO6 la procedura è del tutto analoga tenendo conto però che in questo caso il gestore delle risorse è LSF.

6 Codici di calcolo utilizzando tecniche di deep learning

L'Intelligenza Artificiale (IA) rappresenta un elemento cruciale nella Scienza dei Materiali, consentendo previsioni accurate riguardo lo studio dei materiali cristallini tramite l'analisi dei dati. Nel progetto IEMAP i dati sono conservati nel database di progetto e sono generati dai codici di modellistica atomistica, descritti nelle sezioni precedenti, e dai laboratori sperimentali.

Un esempio di modello di apprendimento automatico applicato ai materiali cristallini è GeoCGNN (Jiucheng Cheng, 2021), acronimo di Geometric-Information-Enhanced Crystal Graph Neural Network. Questo modello è stato sviluppato da Cheng Jiucheng, Lifeng Dong e Chunkai Zhang nel 2020 e si è dimostrato altamente efficace nella previsione delle proprietà dei materiali cristallini. Tale modello risulta essere migliore del 30,3%, del 14,6% e del 13% rispetto ai modelli CGCNN, MEGNet e iCGCNN, rispettivamente (Grossman, 2018).

In virtù della sua alta accuratezza delle previsioni, nel nostro lavoro è stata adottata GeoCGNN per gli obiettivi del progetto IEMAP. Il suo codice è open-source è stato messo a disposizione dagli autori nella repository GitHub: https://github.com/Tinystormjojo/geo-CGNN.
Sulla base di questa libreria è stato prodotto il codice GitLab ENEA che integra tale libreria con il workflow AiiDA, disponibile al link: https://gitlab.brindisi.enea.it/claudio.ronchetti/ai4mat, ed un codice esclusivo per l'utilizzo della rete GeoCGNN: https://gitlab.brindisi.enea.it/marco.catillo/geocgnn-matenergy.

Quest'ultimo implementa la k-fold-cross validation ed è stata programmata in due versioni, una che elabora una struttura cristallina ed un'altra che invece ne elabora due per predirne una data quantità associata. La prima è adatta per la predizione, per esempio, di quantità quali energia di formazione, band gap, energy above hull e molte altre che sono associate ad un singolo cristallo. La seconda invece è utile a stimare quantità come il potenziale redox che dipende da due cristalli distinti, poiché associata alla reazione di intercalazione che porta un cristallo scarico ad uno carico o viceversa.
Inoltre, la libreria geocgnn-matenergy è ottimizzata per essere utilizzata all'interno dell'infrastruttura CRESCO, oltre a poter essere utilizzata sul proprio personal computer. Ne descriviamo nelle prossime sezioni i dettagli di utilizzo.

6.1 Ambiente virtuale CONDA

6.1.1 Utilizzo all'interno di CRESCO

Nei diversi cluster CRESCO sono stati predisposti dei contenitori CONDA che permettono di mettere assieme le librerie di PyTorch, pymatgen, mpi4py e molte altre in modo compatibile tra loro e insieme con la versione di CUDA disponibile nei vari nodi di CRESCO. Questi contenitori servono a creare il giusto setup per lanciare geocgnn-matenergy.
Tale codice è stato testato sulle risorse computazionali di ENEAGRID provvisti di GPU NVIDIA per studiarne le sue performance nella fase di training e inferenza. È importante sottolineare che GeoCGNN, come l'implementazione della maggior parte delle reti neurali basate su PyTorch, traggono vantaggio dall'uso delle GPU NVIDIA che permettono di accelerare notevolmente la computazione rispetto al solo utilizzo della CPU. È quindi interessante dunque fare test sui nodi provvisti di GPU.
I cluster di CRESCO che hanno nodi con GPU NVIDIA sono: CRESCO6 con 2 nodi (nvi1 e nvi2) con GPU TESLA V100; CRESCO5F con 4 nodi (001-004) con GPU TESLA A100 e 1 nodo (011) con GPU NVIDIA H100; inoltre, abbiamo CRESCOB costituito da un singolo nodo che possiede una GPU Quadro RTX 6000. Recentemente il cluster XCRESCO è entrato in funzione con 60 nodi provvisti di 4 GPU TESLA V100 per nodo, mentre il cluster CRESCO5F è stato dismesso. I cluster CRESCO7 e molti riportati in sezione 2, non posseggono GPU NVIDIA, per cui l'utilizzo della rete GeoCGNN su questi cluster non è stata presa in considerazione. In Tabella 1, sono riassunte le caratteristiche delle GPU presenti su CRESCO.

Tabella 1. Lista GPU NVIDIA disponibili in ENEAGRID.

CRESCO	GPU NVIDIA	Versione CUDA
CRESCO5F x[001-004]	Tesla A100-PCIE-40GB	12.2
CRESCO5F x011	H100 PCIe	12.2
CRESCOB	Quadro RTX 6000	11.5
CRESCO6 [nvi1,nvi2]	Tesla V100-PCIE-32GB	10.1
XCRESCO x[001-060]	Tesla V100-SXM2-16GB	12.0

Inoltre, ogni cluster ha dei suoi moduli disponibili che possono essere caricati dal singolo utente. In base, dunque, al cluster su cui si vuole lavorare i moduli da caricare possono cambiare, come anche la versione di Python da utilizzare. È quindi necessario un ambiente CONDA opportuno per le varie casistiche di utilizzo. In Tabella 2 riportiamo pertanto diversi setup necessari (moduli da caricare e ambiente CONDA da usare) per il funzionamento di GeoCGNN in base a quale cluster con GPU si sta utilizzando.

Tabella 2. Lista ambienti CONDA e moduli da caricare per utilizzare GeoCGNN su CRESCO.

CLUSTER	CONDA	Moduli
CRESCO5F x[001-004]	ai4matenv	base gcc/gcc(default) pgi/pgi_20.4 intel/intel20(default) lsf10.1 pfs nvhpc-22.3
CRESCO5F x011	ai4matenv	base pgi/pgi_19.7 gcc/gcc730 lsf10.1 intel/intel17 mpi_flavour/impi-intel17 pfs intel/intel20
CRESCOB	ai4matenv6nvi1	base gcc/gcc(default) lsf intel pgi/pgi11_10(default) mpi_flavour/openmpi_intel-1.2.8(default)
CRESCO6 [nvi1,nvi2]	ai4matenv6nvi1	base pgi/pgi_20.4(default) gcc/gcc730(default) lsf10.1 intel/intel17(default) mpi_flavour/openmpi_intel17-3.1.2(default) pfs
XCRESCO x[001-060]	ai4matenvx	base cuda/nvhpc/24.1 blas/3.12.0--cuda-nvhpc--24.1 openmpi/4.1.6--gnu--11.3--cuda profile/phys qiskit/1.1.1

Diversi cluster richiedono anche diversi applicativi CONDA, in base alla architettura utilizzata dai nodi. Per esempio, XCRESCO ha nodi con architettura ppc64le, mentre per CRESCO6, CRESCO5F e CRESCOB, l'architettura è x86_64. A tal fine sono stati messi a disposizione due eseguibili di CONDA, disponibili nei seguenti percorsi:

- */afs/enea.it/project/ai4mat/software/miniconda3/bin/conda* => per CRESCO5F, CRESCO6 e CRESCOB;
- */afs/enea.it/project/ai4mat/software/miniforge3/bin/conda* => per XCRESCO.

Come possiamo notare, crearsi di volta in volta l'ambiente ottimale in base alla macchina che si sta utilizzando può quindi risultare abbastanza complesso. Per questo motivo è stato preparato uno script che riconosce la macchina che si sta utilizzando, fa il loading dei moduli necessari secondo la Tabella 2, utilizza la versione opportuna di CONDA e attiva l'ambiente CONDA necessario. Lo script può essere lanciato con il comando da terminale:

```
source /afs/enea.it/project/ai4mat/software/cresco_env_setup.sh
```

Gli ambienti CONDA che andrà a scegliere saranno in base alla macchina utilizzata i seguenti:

- *ai4matenvx* => per XCRESCO;
- *ai4matenv* => per CRESCO5F;
- *ai4matenv6nvi1* => per CRESCO6 e CRESCOB.

Essi sono disponibili nel percorso */afs/enea.it/project/ai4mat/software/*, nel caso si vogliano clonare nella propria area personale per usarli per i propri progetti specifici.

6.1.2 Utilizzo sul proprio pc

Per l'utilizzo sul proprio pc, è disponibile il file ai4matenvx.*yml* disponibile all'interno del repository di GeoCGNN: https://gitlab.brindisi.enea.it/marco.catillo/geocgnn-matenergy. Essa è testata per il caso di una macchina con GPU V100 e versione di CUDA 12.0. La creazione e l'attivazione dell'ambiente di CONDA può essere fatto con i seguenti comandi da terminale:

```
git clone git@gitlab.brindisi.enea.it:marco.catillo/geocgnn-matenergy.git
cd geocgnn-matenergy/
conda env create -f ai4matenvx.yml
conda activate ai4matenvx
```

Con tali comandi il repository GitLab viene copiato nella propria macchina e l'ambiente viene creato e attivato. È importante notare che nel caso di una versione diversa di CUDA, la versione opportuna di PyTorch, pymatgen, mpi4py etc. devono essere trovate autonomamente in modo opportuno.

6.2 Quick start

Con la rete GeoCGNN in https://gitlab.brindisi.enea.it/marco.catillo/geocgnn-matenergy, si può fare sia training che inferenza. A tale scopo, la rete prende come input il file CIF (Crystallographic Information File) della struttura da studiare. Tale file contiene l'intera informazione sulla struttura del cristallo con le relative posizioni degli atomi che lo compongono. I file CIF possono essere scaricati da database liberamente disponibili sul web come, ad esempio, Materials Project (MP). Le REST API di MP per estrarre i file CIF possono

essere consultate sul sito: https://api.materialsproject.org/docs. Una volta fatto il download dei CIF da studiare, il training e l'inferenza sulle macchine CRESCO sono descritte nelle successive sottosezioni.

6.2.1 Training e validazione

Per il training e validazione, mettiamo tutti i file CIF da cui studiare le proprietà dei rispettivi materiali in una data cartella. A questo punto bisogna preparare un file di input dove ci sono tutte le informazioni su dove andare a prendere i file CIF, dove stampare i file di output, il tipo di algoritmo e tutti i parametri della rete che si vogliono utilizzare. Per le informazioni sul significato dei parametri della rete GeoCGNN rimandiamo a Ref. [Jiucheng Cheng, C. Z. (2021)].

Preparazione file di input

All'interno di https://gitlab.brindisi.enea.it/marco.catillo/geocgnn-matenergy è presente il file *main.in* come esempio di come scrivere un file di input. Esso consiste di quattro sezioni:

1. *[ALG]* # specifica quale algoritmo utilizzare tra le due opzioni disponibili: single o double;
2. *[CUDA]* # specifica se utilizzare la GPU oppure no, in tal caso verrà utilizzata la CPU;
3. *[ATOM_GRAPH]* # sezione per la conversione dei file CIF in grafi che vengono salvati in file npz;
4. *[PROCESS_geoCGNN]* # sezione che contiene i parametri della rete con cui fare il training.

Vediamo ora nello specifico queste quattro sezioni. Ognuna di esse ha diverse istanze che vanno specificate e che descriviamo con un commento preceduto dal simbolo #.

1. ALG

```
[ALG]
type single # specifica se predire una quantità da un singolo CIF o da una coppia di CIF, in questo ultimo caso va scritto double
```

2. CUDA

```
[CUDA]
Enable true # se true attiva l'uso della GPU, se false utilizza la CPU
Deterministic true # se true si utlizza l'algoritmo deterministico di cuBLAS per la GPU, se false quello non deterministico
```

3. ATOM_GRAPH

```
[ATOM_GRAPH]
activate true # se true è attivo, esegue la conversione da CIF a grafi
cif_dir ./test/database/cif # cartella dove va a leggere i CIF da convertire
npz_dir ./test/database/npz_single # cartella dove scrivere il file dei grafi
name_database MP # prefisso che si vuole dare ai file npz
base_name mp- # prefisso iniziale dei file CIF
onehot_file ./test/database/config_onehot.json # file di output dove sono scritte le informazioni di conversione da CIF a grafo
cutoff 8 # cutoff in Angstrom per la costruzione del grafo
max_num_nbr 12 # numero di atomi primi vicini da considerare per la costruzione del grafo
compress_ratio 1 # frazione di CIF contenuti in cif_dir da trasformare in grafo
chunk_size 10000 # quanti CIF elaborare ogni volta per la scrittura dei grafi
```

La sezione di ATOM_GRAPH è essenziale per il training e inferenza. Essa produce due tipi di file di output:

- my_graph_data_MP_8_12_100_<indice del file npz>.npz
- config_onehot.json

Il primo è una serie di file output dove vengono scritti i grafi tradotti dai CIF. Il numero di questi file output equivale ad uno più il numero di file CIF dati nella cartella *cif_dir* diviso *chunk_size*. Questi file sono salvati nella cartella *npz_dir*. Tale cartella è importante che sia stata creata prima dell'avvio del codice. Il secondo è un file output (onehot encoding) che da l'istruzione di come leggere i file npz, se già esiste non viene creato e viene considerato quello preesistente. Se questa sezione del file input è già stata effettuate precedentemente e si è interessati direttamente al training, può anche essere non considerata ponendo *activate->false*.

4. PROCESS_geoCGNN

```
[PROCESS_geoCGNN]
activate            true     # se true il training viene effettuato altrimenti no
n_hidden_feat       128      # numero di feature per la rappresentazione dei nodi
conv_bias           False    # se false il termine bias nell'aggiornamento dei nodi è posto a zero
n_GCN_feat          192      # numero di feature per il pooling
N_block             5        # numero di blocchi convoluzionali
cutoff              8        # cutoff in Angstrom per la lettura del grafo
max_nei             12       # numero di atomi primi vicini da considerare per la lettura del grafo
n_MLP_LR            5        # numero di layer dell'MLP finale
n_grid_K            4        # griglia punti K utilizzata per l'attention mask
n_Gaussian          64       # numero di elementi di base per l'attention mask
node_activation     Sigmoid  # funzione di attivazione tra quelle di PyTorch per l'aggiornamento dei nodi
MLP_activation      Elu      # funzione di attivazione tra quelle di PyTorch per l'MLP
use_node_batch_norm True     # se True utilizza BatchNorm1d di PyTorch sui nodi
use_edge_batch_norm True     # se True utilizza BatchNorm1d di PyTorch sugli edge
batch_size          64       # sottoinsieme di grafi che vengono utilizzati di volta in volta per l'ottimizzazione
optim               adam     # ottimizzatore utilizzato per l'aggiornamento dei pesi
lr                  1e-3     # learning rate
test_ratio          0.2      # frazione dei dati usati come test set per la sola inferenza
clip_value          0        # parametro di clip_grad_value_ di PyTorch per il gradiente utilizzato
milestones          250      # quando il numero di epoche raggiunge questo numero, lr viene diminuito di un
fattore 10
gamma               0.1      # parametro eta_min di CosineAnnealingLR di PyTorch
cosine_annealing    false    # se true utilizza CosineAnnealingLR di PyTorch per l'ottimizzazione
num_epochs          300      # numero di epoche per il training
dataset_path        ./test/database                    # cartella dove si trova il file dei target
history_file        ./test/data/history_{}_k{}.csv     # file di output dove viene scritto MSE e MAE per epoca
model_pth           ./test/model/model_{}_k{}.pth      # file dei pesi generati alla fine del training
test_predictions ./test/data/test_predictions_{}_k{}.csv # file delle predizioni ottenute sul test set
target_file         ./test/database/targets_{}.csv     # file dei target da confrontare con le predizioni
npz_file            ./test/database/npz_single/my_graph_data_MP_8_12_100_{}.npz   # file dei grafi
database            ex_single        # nome del file dei target
```

```
target_name     formation_energy_per_atom    # colonna da predire specificate nel file di target_file
k_val           5          # numero di sottoinsiemi dei dati per il k-fold-cross validation
k_min           0          # indice del sottoinsieme iniziale da considerare nel k-fold-cross validation
num_workers     0          # parametro di PyTorch per il DataLoader
seed            12345      # seme per fare lo splitting pseudo-casuale dei dati per training, validazione e test
load_model      false # se true carica dei pesi iniziali dati in pre_trained_model, se false parte da pesi random
pred            false      # se true non fa il training ma solo inferenza
pre_trained_model_path ./test/pre_trained/model.pth          # file dei pesi con cui inizializzare la rete
```

La sezione *[PROCESS_geoCGNN]* è responsabile del training effettivo. Le parentesi graffe aperte e chiuse che ivi appaiono, sono dei segnaposto che indicano al programma dove scrivere determinate stringhe. Quando questa sezione è attiva, è importante che i file npz nella cartella *npz_single* esistano, poiché essi sono i grafi che vengono letti per fare il training, la validazione o il test. Inoltre, è importante che ci sia il file dei target (in formato CSV), poiché la rete, sul sottoinsieme dei CIF considerati per il training, confronta ad ogni epoca la propria predizione con i valori "veri", tipicamente presi da un calcolo DFT; calcola la loss function, che nel nostro caso è l'errore quadratico medio (MSE) e usa l'ottimizzatore *adam* o *SGD* per modificare i pesi della rete. Per l'input file dato sopra, il target file avrà come nome: *targets_ex_single.csv*, ovvero al posto delle parentesi {} viene sostituita la stringa data da *database*. La sua struttura è quella di un CSV con almeno due colonne separate da una virgola: *id* del materiale *e formation_energy_per_atom*. Quest'ultimo è specificato dall'istanza *target_name* nell'input file. Quindi *targets_ex_single.csv* avrà, ad esempio, la forma:

```
id,average_voltage,formation_energy_per_atom,battery_formula
mp-764034,3.8420038315517253,-1.5417107294827577,Na0-0.5FeO2
mp-560242,3.787743999051724,-2.9654893933620685,Na0-1MnF3
mp-1221482,1.3885208965517293,-1.120416987443181,Na0.5-0.67NbSe2
```

in cui l'ordine delle colonne non è importante. Nel caso in cui abbiamo *[ALG]->type->double*, dovremmo avere tre colonne: *id1*, *id2* e il nome specificato in *target_name*. Prendiamo ad esempio *target_name->average_voltage* avremmo allora che il file target sarà ad esempio:

```
id1,id2,average_voltage,formation_energy_per_atom,battery_formula
mp-764034,mp-1395285,3.8420038315517253,-1.5417107294827577,Na0-0.5FeO2
mp-560242,mp-1176437,3.787743999051724,-2.9654893933620685,Na0-1MnF3
mp-1221482,mp-1221396,1.3885208965517293,-1.120416987443181,Na0.5-0.67NbSe2
```

Questo perché l'average voltage (o anche potenziale redox) è una quantità che può essere meglio predetta se consideriamo sia il materiale carico che scarico nella reazione di intercalazione in una batteria.
Un altro file di input che viene specificato in *[PROCESS_geoCGNN]* è il *pre_trained_model.* Questo è richiesto se *load_model* è posto a *true*. In tal caso la rete viene inizializzata con i pesi specificati in questo file anziché considerare pesi random.

File di output

Durante il training il file CSV specificato in *history_file* viene creato. Esso contiene i valori per ogni epoca della loss function (MSE) e dell'errore medio assoluto (MAE) calcolati sul set di training e di validazione. Di sotto è riportato un esempio:

```
epoch,train_loss,train_mae,val_loss,val_mae
0,3.8476399122146305,1.8900880324117713,4.730562666091086,2.0641800743555314
1,2.6852825285747377,1.6003860620570056,5.054049919622365,2.136907067202183
2,1.8771967649180932,1.3366789132397316,5.060009785263847,2.1508307936782955
3,1.2261519882734302,1.0750350037799938,3.5422217046228845,1.8046610333060327
```

Al posto di {} dell'istanza *history_file* viene posto un numero da *k_min* fino a *k_max*, con *k_max.* Quest'ultimo che definiamo come

$$k_{max} = (k_{val} - 1)\theta(x) + N_{GPU}\theta(-x),$$

con $x = N_{GPU} - k\_val + 1 - k\_min$, N_{GPU} il numero di GPU eventualmente disponibili nel nodo e θ la funzione di Heaviside. Questo è dovuto al k-fold-cross validation. Nel nostro caso vengono performati *k_max-k_min* training in parallelo, ovvero ognuno assegnato ad una GPU, nel caso in cui la sezione *[CUDA]* è attiva e sono presenti GPU sul nodo. Se questo non è il caso, il training è performato su CPU e *k_max=k_val*-1. Questo perché ogni training viene eseguito su un diverso training set estratto col metodo del k-fold-cross validation.

Alla fine del training altri file di output vengono creati. Il primo è un file CSV specificato dall'istanza *test_predictions*, dove viene riportato sul test set il valore delle predizioni insieme ai valori target. Nel caso in cui *[ALG]* è posto a *single*, abbiamo la seguente struttura:

```
name,prediction,target
mp-1233258,-2.5647671,-2.5856404
mp-1234190,-1.4653854,-1.4047642
mp-1234486,-2.8165886,-2.7994728
```

dove in *name* abbiamo l'id del materiale. Nel caso in cui *[ALG]* è posto a *double*, abbiamo invece:

```
name1,name2,prediction,target
mp-780565,mp-779142,3.3272614,3.7095938
mp-1104133,mp-21021,1.5023352,1.6752464
mp-771204,mp-772135,3.6227322,3.868106
```

dove *name*1 e *name*2 sono gli id dei due materiali coinvolti nella predizione di una data quantità. Anche in questo caso nelle parentesi graffe viene riportato un numero da *k_min* a *k_max* e alcuni parametri della rete usati, specificatamente *N_block*, *cutoff* e *max_nei*.

Per finire, vengono generati *k_max-k_min* file dei pesi finali con estensione *pth*. Questi possono essere eventualmente riutilizzati per ulteriori training come inizializzazione dei pesi o per fare solo inferenza. Il nome di questi file è specificato in *model_pth* dove al posto delle parentesi graffe viene sostituita una stringa contenente l'informazione su *N_block*, *cutoff* e *max_nei* usati e nelle parentesi graffe finali viene posto un numero da *k_min* a *k_max* corrispondente al training set utilizzato.

6.2.2 Sottomissione Job

Una volta che il file *main.in* è stato scritto opportunamente si può lanciare il training in due modi. Il primo è in interattivo sulla propria macchina o su un nodo di CRESCO dove si è autorizzati a lanciare programmi in questa modalità. Esempi di tali nodi sono quelli di front-end quali: *crescob-nvi1.brindisi.enea.it*, *xcrescox002.frascati.enea.it* o anche *cresco6x002.portici.enea.it*; in CRESCO6 nei nodi *cresco6-nvi1.portici.enea.it* e *cresco6-nvi2.portici.enea.it*. In interattivo il job può essere lanciato come

```
mpirun -n 4 python3 /afs/enea.it/project/ai4mat/software/geocgnn-matenergy/main.py -i main.in
```

il quale lancia il programma su quattro processi MPI. È importante specificare che, quando la sezione *[PROCESS_geoCGNN]* è attiva, il numero di processi da utilizzare deve coincidere col numero di GPU montate sul nodo. Ad esempio, su XCRESCO dove tutti i nodi hanno 4 GPU V100 montate, possiamo dare l'opzione *-n 4*, mentre ad esempio sul nodo *crescob-nvi1.brindisi.enea.it* abbiamo una sola GPU RTX 6000 montata, in questo caso dovremmo usare l'opzione *-n 1*.
Questo discorso si estende anche quando lanciamo il programma attraverso un batch scheduler. Riportiamo un esempio di job script che può essere mandato nei sistemi provvisti di SLURM, come XCRESCO:

```
#!/usr/bin/pagsh

#SBATCH --job-name=prova
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4
#SBATCH --time=23:59:59
#SBATCH --out=out/%x.%j.out.dat
#SBATCH --err=out/%x.%j.out.err
#SBATCH --partition=xcresco
#SBATCH --qos=normal
#SBATCH --auks=yes

aklog

PYPATH=/afs/enea.it/project/ai4mat/software/geocgnn-matenergy

mpirun -n $SLURM_NTASKS python3 $PYPATH/main.py -i main.in
```

mentre sui sistemi con LSF, come CRESCO6, un esempio di job script è dato da

```
#!/bin/sh

#BSUB -J prova
#BSUB -W 23:59
#BSUB -n 4
#BSUB -o prova.out.lsf.dat
#BSUB -e prova.out.err.dat
#BSUB -q cresco6_h144

PYPATH=/afs/enea.it/project/ai4mat/software/geocgnn-matenergy
N_procs=`cat $LSB_DJOB_HOSTFILE | wc -l`

mpirun -n $N_procs python3 $PYPATH/main.py -i main.in > main.out
```

In questo ultimo caso, il job utilizzerà solo CPU poiché i nodi di calcolo di CRESCO6 non hanno GPU. Ulteriori informazioni su come lanciare job script si possono consultare sulla pagina web di ENEA di CRESCO: https://www.eneagrid.enea.it/Resources_en/CRESCO_documents/index.html.

6.2.3 Inferenza

L'inferenza può essere fatta in due modi. Il primo è di specificare nel file *main.in* di input, *test_ratio->1.0*, *load_model->true* e *pred->true* e specificando in *pre_trained_model_path* i pesi da porre nella rete. A questo punto si può lanciare il programma *main.py* nelle modalità descritte nella precedente sezione. In alternativa, si può importare in un proprio codice Python la routine *inference.py* per la sola inferenza. Essa si trova nel percorso su CRESCO: */afs/enea.it/project/ai4mat/software/geocgnn-matenergy/*, sia per l'algoritmo *single* che *double*.
Diamo si seguito uno snapshot di codice per l'utilizzo:

```
import sys
import pprint
sys.path.insert(0,"/afs/enea.it/project/ai4mat/software/geocgnn-matenergy/src/")

from single_geocgnn.inference import prediction, Input
args=Input(
    cif="./test/database/cif/mp-1395285.cif",
    pth="./test/pre_trained/model_5_8_12_0.pth",
    onehot="./test/database/config_onehot.json")
out=prediction(args)
pprint.pprint(out)

from double_geocgnn.inference import prediction, Input
args=Input(
    cif1="./test/database/cif/mp-764034.cif",
    cif2="./test/database/cif/mp-1395285.cif",
```

```
    pth="./test/pre_trained/model_5_8_12_0_d.pth",
    onehot="./test/database/d_config_onehot.json")
out=prediction(args)
pprint.pprint(out)
```

La classe *Input* per l'algoritmo *single* definisce l'input da utilizzare e prende come argomenti, il *cif* della struttura, i pesi della rete *pth*, e il file di *onehot* encoding precedentemente generato dalla sezione *[ATOM_GRAPH]*. Per l'algoritmo *double* i parametri da passare in Input sono pressoché gli stessi, tranne per il fatto che bisogna specificare due file cif: *cif1* e *cif2*. La funzione *prediction* che prende come argomento un oggetto *Input* esegue la predizione sulla GPU se presente, altrimenti sulla CPU, e ritorna un dizionario con la predizione. La quantità da predire viene dedotta dai pesi che vengono dati, non c'è bisogno quindi di specificarla. Il risultato del precedente snapshot è il seguente:

```
{'cif_name': 'mp-1395285',
 'formation_energy_per_atom': {'prediction': -0.016459068283438683},
 'time_prediction(s)': 0.6145009994506836}
{'average_voltage': {'prediction': 0.007883528247475624},
 'cif1_name': 'mp-764034',
 'cif2_name': 'mp-1395285',
 'time_prediction(s)': 0.031432151794433594}
```

Il dizionario di output contiene dunque l'id (o gli id) del (dei) materiali, il valore della predizione, cosa si sta predicendo e infine il tempo impiegato per la predizione.

Una lista di modelli e file di onehot encoding, sono disponibili per la predizione dell'energia di formazione per atomo e potenziale redox (o average voltage), essi sono reperibili nei seguenti percorsi:

- */afs/enea.it/project/ai4mat/software/geocgnn-matenergy/models/formation_energy_per_atom/*
- */afs/enea.it/project/ai4mat/software/geocgnn-matenergy/models/redox_potential_single/*
- */afs/enea.it/project/ai4mat/software/geocgnn-matenergy/models/redox_potential_double/*

In tutte le seguenti cartelle sono disponibili 15 file *pth* per i pesi della rete e un file *onehot.json* per il onehot encoding. La differenza tra le cartelle *redox_potential_single* e *redox_potential_double* è che nella prima i pesi sono stati generati prendendo come target solo i cif del materiale scarico della reazione di intercalazione della batteria, mentre nella seconda cartella sono stati generati da entrambi i cif del materiale scarico e carico. Questi ultimi pesi daranno una predizione leggermente più accurata del caso *single*, in quanto l'algoritmo GeoCGNN *double* che li genera è più aderente alla reazione di intercalazione che prende luogo nella batteria.

6.3 Benchmark su CRESCO

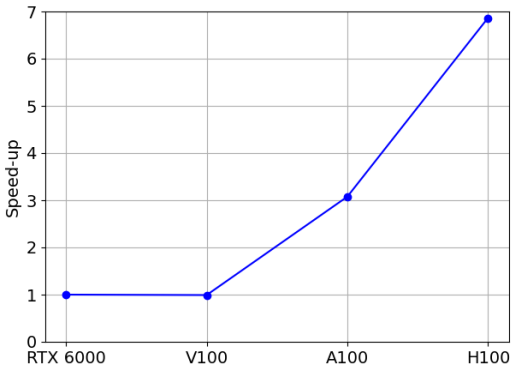
In sezione 6.1 abbiamo visto come gestire gli ambienti CONDA all'interno dei vari cluster e nodi disponibili su CRESCO. In base alla GPU e al sistema utilizzato, il codice GeoCGNN può differire anche di molto nella performance, del training o inferenza. Abbiamo quindi studiato come varia lo speed-up in base alle diverse GPU in uso in CRESCO, ovvero RTX 6000, V100, A100 e H100. A tal fine abbiamo preso circa 130k CIF da

Materials Project per il training sull'energia di formazione per atomo per 300 epoche e calcolato lo speed-up rispetto alla GPU RTX-6000, così definito:

$$\text{Speed-up}_i = \frac{t_{RTX6000}}{t_i},$$

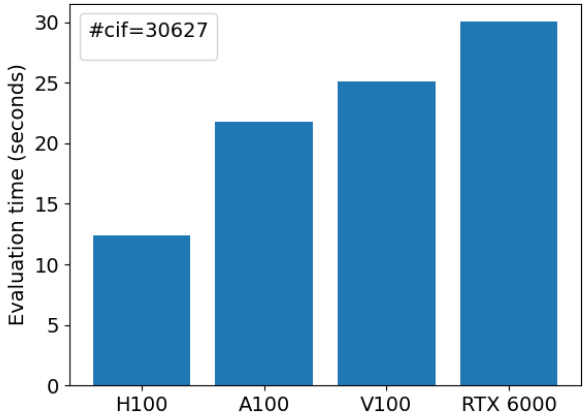
dove t_i è il tempo impiegato per il training dalla GPU *i*.

Figura 1. Speed-up relativo al training di GeoCGNN sull'energia di formazione per diverse GPU.



In Figura 1, riportiamo i valori di speed-up ottenuti. Come è evidente, le GPU H100 sono 7 volte più veloci rispetto la RTX6000 e più del doppio veloci rispetto alle A100. Tra RTX6000 e V100 non si riscontrano però particolari differenze. Vediamo ora in Figura 2 il tempo totale impiegato per l'inferenza su 30k CIF di Materials Project. Il programma in questo caso ha già caricato i pesi della rete neurale ed ha quindi soltanto calcolato il valore dell'energia di formazione ai valori ottimizzati dei pesi della rete. Notiamo ancora una volta come l'H100 risulta essere la più veloce, mentre a seguire A100, V100 e RTX6000 sono progressivamente più lente.

Figura 2. Tempo in secondi impiegato per l'inferenza dell'energia di formazione al variare delle GPU usate.



L'H100 sembra quindi essere la più performante sia in fase di training che di inferenza. Abbiamo poi controllato i tempi per la predizione del potenziale di intercalazione (Tabella 3) per due diverse

implementazioni di GeoCGNN, che prende uno o due CIF alla volta, essendo il potenziale redox una quantità connessa con due materiali, carico e scarico.
Il training per il potenziale redox è stato fatto su 3k materiali catodici riportati su Materials Project e il training è stato performato su 300 epoche. Una volta che il training è terminato, l'inferenza è stata fatta su 866 materiali catodici. Entrambi questi calcoli sono stati fatti sulle GPU V100 del cluster XCRESCO.

Tabella 3. Tempi di training e inferenza per potenziale di intercalazione su GPU NVIDIA V100.

Approccio	Training	Inferenza singolo potenziale redox
Potenziale da singolo CIF	29 m	8.9×10^{-4}s
Potenziale da due CIF	55 m	1.5×10^{-3} s

Come si può osservare la valutazione di una singola quantità prende un tempo dell'ordine di millesimi di secondo, mentre il training meno di un'ora. I tempi sono quindi decisamente più piccoli rispetto ai metodi tradizionali basati su software DFT come Quantum ESPRESSO, dove la simulazione di un singolo materiale può durare diverse ore.

7 Conclusioni

In questo documento abbiamo riassunto l'attività svolta nel terzo anno di finanziamento del progetto IEMAP riguardante la LA1.3. L'attività si è articolata in modo tale da fornire un utilizzo efficiente dell'infrastruttura di supercalcolo dell'ENEA per la piattaforma IEMAP. Con questa finalità si provveduto ad aggiornare e rendere disponibili i codici di calcolo sulle nuove risorse computazionali rendendoli disponibili in modo dedicato agli utenti della piattaforma IEMAP e definendo modalità di utilizzo semplificate

8 Abbreviazioni ed acronimi

ACFDT-RPA = Adiabatic Connection Fluctuation Dissipation Theorem – Random Phase Approximation
ACL = Access Control List
AFS = Andrew File System
AiiDA = Automated Interactive Infrastructure and Database for computational science
AlmaLinux = distribuzione Linux libera e open-source sviluppata da AlmaLinux OS Foundation
AM1 = Austin Model 1
AMBER = Assisted Model Building with Energy Refinement
BLAS = Basic Linear Algebra Subroutines
BMC = Baseboard management controller
CentOS = Community enterprise Operating System
CGCNN = Crystal Graph Convolutional Neural Network
CHARMM = Chemistry at Harvard Macromolecular Mechanics
CIF = Crystallographic Information File
CONDA = pacchetto open-source di gestione di ambienti per Python
CP = Car-Parrinello pacchetto software per dinamica molecolare
CP2K = Car-Parrinello pacchetto software
CPU = Central Processing Unit
CRESCO = Computational RESearch centre on COmplex systems

CSV = Comma Separated Values
cuBLAS = Basic Linear Algebra Subroutines per CUDA
CUDA = Compute Unified Device Architecture
DFT = Density Functional Theory
DFTB = Density Functional Based Tight Binding
ENEAGRID = infrastruttura delle risorse computazionali di ENEA gestite dalla divisione ICT
EPW = pacchetto per il calcolo elettrone-fonone che usa funzioni di Wannier
GbE = Gigabit Ethernet
GCC = GNU Compiler Collection
GCN = Graph Convolutional Network
GeoCGNN = Geometric-Information-Enhanced Crystal Graph Neural Network
GGA = Generalized Gradient Approximation
Git = software per il controllo di versione
GitHub = piattaforma web open-source per la gestione di repository Git
GitLab = piattaforma web open-source per la gestione di repository Git
GNU = GNU's Not Unix
GPFS = General Parallel File System
GPL = General Public License
GPU = Graphics Processing Unit
GW = metodo Green's function e interazione di Coulomb W
GWL = codice open-source di Quantum ESPRESSO per performare calcoli GW con approssimazione G_0W_0
HPC = High-Performance Computing
IA = Intelligenza Artificiale
IBM = International Business Machines Corporation
iCGCNN = Improved Crystal Graph Convolutional Neural Network
IEMAP = Italian Energy Materials Acceleration Platform
IPMI = Intelligent Platform Management Interface
JSON = JavaScript Object Notation
LA = Linea di Attività
LAMMPS = Large-scale Atomic/Molecular Massively Parallel Simulator
LDA = Local Density Approximation
LSF = Load Sharing Facility
LUSTRE = sistema open-source di file parallelo distribuito
MAE = Mean Absolute Error
MEGNet = MatErials Graph Network
Mellanox-EDR = sistema di collegamento rete sviluppato da Mellanox Technologies, oggi acquisita da NVIDIA
MKL = Math Kernel Library
MLP = Multi Layer Perceptron
MNDO = Modified Neglect of Diatomic Overlap
MP = Materials Project
MP2 = Møller–Plesset perturbation theory al secondo ordine
MPI = Message Passing Interface
mpi4py = Wrapper MPI per Python

MSE = Mean Squared Error
NEB = Nudged Elastic Ban
NFS = Network File System
NPZ = Numpy's compressed array format
NVHPC = suite di compilatori di NVIDIA per GPU
Omni-Path = architettura di comunicazione ad alte prestazioni sviluppata da Intel
OpenAFS = Implementazione open-source di AFS
OpenMP = Open Multi Processing
OST = Object Storage Target
PGI = Portland Group Inc.
PHonon = pacchetto open-source di Quantum ESPRESSO per lo studio dell'interazione elettrone-fonone
PM3 = Parametric Method numero 3
PM6 = Parametric Method numero 6
PostProc = pacchetto open-source di Quantum ESPRESSO per il post-processing dei dati prodotti da PWscf
PTH = Python Pickled Object
PWcond = programma open-source di Quantum ESPRESSO per computare la trasmittanza di un sistema
PWneb = Plane-Wave Nudged Elastic Ban
PWscf = Plane-Wave Self-Consistent Field
Pymatgen = Python Materials Genomics software
PyTorch = libreria per il Machine Learning basata su Torch
QE = Quantum ESPRESSO
qiskit = Quantum Information Software Kit
QM/MM = Quantum Mechanics/Molecular Mechanics
RAM = Random Access Memory
RM1 = Recife Model 1
RMM-DIIS = Residual Minimization Method with Direct Inversion in the Iterative Subspace
Round-robin = algoritmo per il network scheduling
RPA = Random Phase Approximation
SATA = Serial Advanced Technology Attachment
SCF = Self Consistent Field
SLURM = Simple Linux Utility for Resource Management
TDDFPT = Time-Dependent Density-Functional Perturbation Theory
VASP = Vienna Ab-initio Simulation Package
XSPECTRA = codice open-source di Quantum ESPRESSO per il post-processing dei dati della densità di carica prodotti da PWscf

Bibliografia

Giannozzi, P. e. (2009). QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *J. Phys.: Condens. Matter. 21*, 395502.

Grossman, J. C. (2018). Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Phys. Rev. Lett. 120, 145301*.

Hafner, J. (2007). Materials simulations using VASP—a quantum perspective to materials science. *Computer Physics Communications 177*, 6.

Jiucheng Cheng, C. Z. (2021). A geometric-information-enhanced crystal graph network for predicting properties of materials. *Communications Materials*.

Kühne, T. D. (2020). CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations. *J. Chem. Phys. 152*, 194103.

Thompson, A. P. (2022). LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm. 271*, 10817.