MISSION INNOVATION ITALY

Ministero dell'Ambiente e della Sicurezza Energetica

Caratteristiche di utilizzo dei codici di calcolo per la piattaforma IEMAP

Simone Giusepponi, Filippo Palombi, Claudio Ronchetti, Massimo Celino

ENEA

Report MI21-24/25

D1.2, CARATTERISTICHE DI UTILIZZO DEI CODICI DI CALCOLO PER LA PIATTAFORMA IEMAP

Simone Giusepponi (ENEA), Filippo Palombi (ENEA), Claudio Ronchetti (ENEA), Massimo Celino (ENEA)

Maggio 2023

Report MISSION INNOVATION

Ministero dell'Ambiente e della Sicurezza Energetica - ENEA
Mission Innovation 2021-2024 - II 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 LA1.2 i cui obiettivi definiti nel Piano di Attività sono la predisposizione degli ambienti di accesso e utilizzo dell'infrastruttura HPC dell'ENEA per gli utenti del progetto IEMAP. Inoltre si prevede l'installazione dei codici di calcolo e valutazione delle loro performance tramite benchmark dedicati.

Le attività riportate in questo report sono necessarie e propedeutiche per la realizzazione della piattaforma di progettazione integrata e accelerata dei materiali per l'energia. In particolare, sono stati predisposti quei codici che sono richiesti dagli utenti della piattaforma IEMAP (Piattaforma Italiana Accelerata per i Materiali per l'Energia) e che necessitano dell'utilizzo dell'infrastruttura di calcolo ad alte prestazioni CRESCO e dei servizi ENEAGRID. Codici necessari per lo studio di materiali innovativi per l'energia che sono stati compilati ed installati in ambiente ENEAGRID per un uso efficace sul supercomputer CRESCO6. Questi codici sono stati anche testati su casi d'uso specifico e finalizzati per i materiali di interesse, valutandone le performance di calcolo e di scalabilità parallela. 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. 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. Il supporto fornito è stato indirizzato dalle esigenze specifiche di ciascun utente ed è stato progettato insieme un percorso di utilizzo della piattaforma CRESCO mettendo a disposizione tutti i servizi di cui ha bisogno.
Per ogni utente sono state allocate le risorse necessarie per ottimizzare il lavoro evitando interferenze con gli altri utenti e sono stati installati i codici sull'infrastruttura CRESCO/ENEAGRID, con o senza licenza, necessari all'esecuzione delle attività. L'insieme dei codici della piattaforma IEMAP sono stati installati in una area dedicata in AFS e mantenuti per l'intera durata del progetto. Attraverso le ACL di AFS è stato possibile dare accesso ai codici licenziati ai soli utenti autorizzati ad usarli. La descrizione di questi codici è stata riportata nella Sezione 2.
Sono state organizzate sessioni dedicate per la realizzazione di benchmark dei codici installati per determinare la migliore configurazione, verificandone, nel contempo, la robustezza e l'usabilità. I risultati di questi benchmark sono riportati nella Sezione 3. Nella Sezione 4, vengono brevemente descritti i sistemi oggetto di studio e anticipati alcuni dei risultati scientifici raggiunti. Infine nella sezione 5, sono descritte le attività di implementazione ed utilizzo di librerie di machine learning.

2 Codici di calcolo per modellistica atomistica

In questa sezione verranno elencati e descritti i codici di calcolo ad alto parallelismo presenti in ENEAGRID e utilizzabili sull'infrastruttura di calcolo ad alte prestazioni HPC (High-Performance Computing) CRESCO6. CRESCO6 è l'infrastruttura di calcolo disponibile in ENEAGRID con la maggiore potenza di calcolo. È installato presso il centro di ricerca ENEA di Portici (Na) ed è un sistema di calcolo con potenza di picco pari a 1.4 PetaFlops, costituito da 434 nodi. Ogni nodo ha:

- 2 socket da 24 core con processore Intel Xeon Platinum 8160 con frequenza di clock pari 2.10 GHz e 192 GB di RAM;
- Una interfaccia Intel Omni-Path 100 Gb/s;
- Due interfacce GbE:;
- Supporto BMC/IPMI 1.8 e software per la gestione remota della console.

Si hanno quindi a disposizione 20.832 core connessi tra loro da una rete a larga banda e bassa latenza basata su Intel Omni-Path a 100 Gb/s.
I file system disponibili su CRESCO6 sono:

- AFS (Andrew File System) che è il file system geograficamente distribuito comune a tutta ENEAGRID;

- GPFS (General Parallel File System) che è il file system di IBM ad alte prestazioni per l'I/O parallelo.

Le modalità di accesso a ENEAGRID e in particolare a questa infrastruttura sono descritte nel deliverable D1.1 "Istruzioni per l'accesso, l'autenticazione e l'utilizzo dell'infrastruttura CRESCO". Ulteriori informazioni sono disponibili alla pagina web: www.eneagrid.enea.it/CRESCOportal/.
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.

2.1 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 (Giannozzi, 2009). I codici di basano sulla teoria del funzionale densità (DFT - Density Functional Theory) ed utilizzano la base delle onde piane e pseudopotenziali (www.quantum-espresso.org).
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 di base per il calcolo delle proprietà di struttura elettronica basate sulla teoria funzionale densità con l'utilizzo della base ad onde piane e pseudopotenziali:

- PWscf (PW): *Plane-Wave Self-Consistent Field;*
- CP (CPV): *Car-Parrinello Molecular Dynamics*.

Include inoltre i seguenti pacchetti per compiti più specifici:

- PWneb (NEB): *energy barriers and reaction pathways through the Nudged Elastic Band method;*
- PHonon: *phonons with Density-Functional Perturbation Theory;*
- PostProc (PP): *various utilities for data postprocessing;*
- PWcond: *ballistic conductance;*
- GWL: *GW calculations and solution of the Bethe-Salpeter Equation;*
- XSPECTRA: *K-edge X-ray adsorption spectra;*
- TDDFPT: *calculations of spectra using Time-Dependent Density-Functional Perturbation Theory;*
- EPW: *electron-phonon calculations using Wannier functions;*

Sono inclusi anche i seguenti pacchetti ausiliari:

- PWgui: *a Graphical User Interface, producing input data files for PWscf;*
- atomic: *a program for atomic calculations and generation of pseudopotentials.*

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 attualmente installata in ENEAGRID e che può essere sottomessa a CRESCO6 e la versione 6.7 compilata con compilatori Intel e schema di parallelizzazione MPI (Message Passing Interface).
Quantum ESPRESSO è un'iniziativa aperta realizzata in collaborazione con vari gruppi di ricerca nel mondo coordinati dalla Fondazione Quantum ESPRESSO che include i seguenti membri: la Scuola Internazionale Superiore di Studi Avanzati (SISSA), il Centro Internazionale di Fisica Teorica Abdus Salam (ICTP), il Centro Nazionale di Supercalcolo (CINECA), la Ecole Polytechnique Fédérale de Lausanne (EPFL), l'Oden Institute for Computational Engineering and Sciences e l'University of Texas di Austin.

2.2 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 (Kühne, 2020). 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).
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 attualmente installata in ENEAGRID e che può essere sottomessa a CRESCO6 e la versione 2022.2 compilata con compilatori GNU e schema di parallelizzazione ibrida MPI-OpenMP (Open MultiProcessing).

2.3 VASP

Il codice VASP (Vienna Ab-initio Simulation Package) è un software per il *modelling* 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 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 che è stata compilata con compilatori Intel e schema di parallelizzazione MPI. L'utilizzo di questo software è limitato agli utenti con licenza, mediante la definizione delle ACL (Acces Control List) di AFS.

2.4 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).
Questo codice è scritto in C++ e può essere compilato ed 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 *multi-threading* OpenMP e accelerazione con GPU (Graphics Processing Unit). 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 attualmente installata in ENEAGRID e che può essere sottomessa a CRESCO6 e la versione del 2019 compilata in seriale e parallelo con compilatori Intel e schema di parallelizzazione MPI. LAMMPS è organizzato in "pacchetti", ciascuno con le proprie funzionalità che possono essere installati indipendentemente uno dall'altro. La versione seriale ha inclusi i seguenti pacchetti: ASPHERE, BODY, CLASS2, COLLOID, COMPRESS, CORESHELL, DIPOLE, GRANULAR, KSPACE, MANYBODY, MC, MEAM, MISC, MOLECULE, OPT, PERI, POEMS, PYTHON, QEQ, REPLICA, RIGID, SHOCK, SNAP, SPIN, SRD, VORONOI, USER-BOCS, USER-CGDNA, USER-CGSDK, USER-COLVARS, USER-DIFFRACTION, USER-DPD, USER-DRUDE, USER-EFF, USER-FEP, USER-MANIFOLD, USER-MEAMC, USER-MGPT, USER-MISC, USER-MOLFILE, USER-PTM, USER-QTB, USER-REAXC, USER-SMTBQ, USER-SPH, USER-TALLY, USER-UEF, USER-YAFF. Mentre questi sono quelli disponibili con la versione parallela: ASPHERE, BODY, CLASS2, COLLOID, COMPRESS, CORESHELL, DIPOLE, GRANULAR, KSPACE, MANYBODY, MC, MISC, MOLECULE, MPIIO, OPT, PERI, POEMS, PYTHON, QEQ, REPLICA, RIGID, SHOCK, SNAP, SPIN, SRD, VORONOI, USER-ATC, USER-AWPMD, USER-BOCS, USER-CGDNA, USER-CGSDK, USER-COLVARS, USER-DIFFRACTION, USER-DPD, USER-DRUDE, USER-EFF, USER-FEP, USER-LB, USER-MANIFOLD, USER-MEAMC, USER-MESO, USER-MGPT, USER-MISC, USER-MOFFF, USER-MOLFILE, USER-PTM, USER-QTB, USER-REAXC, USER-SMTBQ, USER-SPH, USER-TALLY, USER-UEF, USER-YAFF.

3 Benchmark dei codici QE, CP2K e VASP

In questa sezione presentiamo ed analizziamo alcuni benchmark relativi ai codici per il calcolo della struttura elettronica e dinamica molecolare quanto-meccanica, Quantum ESPRESSO, CP2K e VASP. Verrà preso in considerazione il tempio di calcolo necessario ad eseguire una specifica tipologia di calcolo per un particolare sistema, al variare delle risorse computazionali richieste, ovvero, il numero di nodi(core) utilizzati. L'infrastruttura di calcolo è CRESCO6 disponibile in ENEAGRID, che è fisicamente situata nel Centro Ricerche ENEA di Portici, ma che risulta accessibile da remoto agli utenti che hanno un account ENEAGRID così come indicato nel deliverable D1.1 "Istruzioni per l'accesso, l'autenticazione e l'utilizzo dell'infrastruttura CRESCO".
Dal valore dei tempi di calcolo in funzione del numero di core utilizzati, si possono ricavare gli andamenti dello speed-up (S) ed efficienza di parallelizzazione (E), grandezze che misurano la scalabilità di un codice parallelo. Detti t_1 (t_s) il tempo di calcolo per la versione seriale del codice, e t_n quello per la versione parallela che utilizza n core, si definisce speed-up il numero

$$S = \frac{t_1}{t_n}.$$

Nel caso ideale in cui i tempi di calcolo siano inversamente proporzionali al numero di core secondo la legge $t_n = t_1/n$, si avrebbe uno speed-up ideale $S = n$. Quindi, l'utilizzo di n core dovrebbe ridurre i tempi di un fattore n rispetto al caso seriale. Ovviamente, essendoci parti di codice non parallelizzabile, e tenendo conto che all'aumentare dei nodi coinvolti nell'esecuzione del calcolo, crescono i tempi di comunicazione tra i nodi, di

accesso alla memoria, di attesa ecc. questo andamento lineare di *S* è atteso fintanto che c'è una notevole riduzione dei tempi di esecuzione legata alla suddivisione del carico di lavoro tra i vari processi.
Si può valutare di quanto ci si discosti dal caso ideale di crescita lineare di *S*, calcolando efficienza di parallelizzazione tramite la relazione

$$E = \frac{S}{n}$$

che può essere espressa anche in termini percentuali. Nel caso ideale, $S = n$, si ha ovviamente un'efficienza pari al 100%.
Questo tipo di test sulla scalabilità parallela è detto *strong scaling mode* in quanto si concentra sull'uso efficiente delle risorse di calcolo per ridurre il tempo di esecuzione di un problema mantenendo costante la dimensione del problema stesso. Il caso in cui la dimensione del problema aumenta all'aumentare del numero delle risorse computazionali, è detto *weak scaling mode*. In questo caso si mantiene costante il carico di lavoro di ciascun nodo(core).
Non si è considerato il tempo di esecuzione per la versione seriale del codice, in quanto i sistemi presi in esame richiedono grande memoria sul singolo nodo e sono quindi non realizzabili sul singolo core. Quindi, abbiamo preso come riferimento il tempo di esecuzione del calcolo utilizzando 1 nodo (48 core) e normalizzando i risultati su questo valore.

3.1 Test di scalabilità di Quantum ESPRESSO

Nell'ambito del progetto IEMAP, uno dei settori di maggiore interesse di tecniche accelerate di progettazione dei materiali è quello delle batterie (WP2). In particolare, risulta necessario trovare validi sostituti ai materiali a base di Litio, elemento chimico largamente utilizzato nei dispositivi attuali e considerato critico. Tra le possibili alternative, si pongono in considerazione i materiali a base di sodio, elemento abbondantemente presente in natura. Inoltre, il sodio presenta un vantaggio significativo rispetto al Litio, in quanto non è soggetto alla reattività estrema con l'ossigeno, fenomeno che può dar luogo a reazioni esplosive all'interno delle batterie.
Come materiale a base di sodio si è scelto quello con composizione chimica $NaMnO_2$. In particolare, per il modello numerico si è considerato un sistema composto da 96 atomi (24 atomi di Na, 24 atomi di Mn e 48 atomi di O). Da questo sistema, che è mostrato nella Figura 1, saranno generate molte varianti nelle quali si sostituiranno atomi di Mn sia con quelli di Ti sia con quelli di Ni; varianti per le quali dovrà essere calcolato lo stato fondamentale. Per definire la migliore richiesta di risorse computazionali, usando come codice Quantum ESPRESSO, si è proceduto a fare un'analisi approfondita della scalabilità parallela di questo codice sull'infrastruttura di calcolo CRESCO6 disponibile in ENEAGRID.

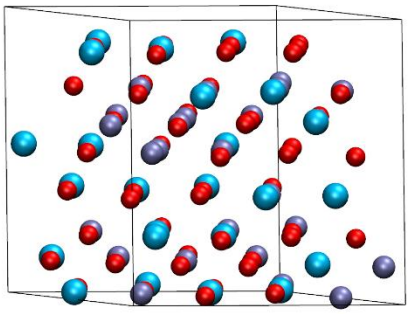

Figura 1. Modello atomistico della cella periodica del cristallo $NaMnO_2$ contenente 96 atomi, sistema utilizzato per il benchmark del codice Quantum ESPRESSO.

Consideriamo il tempo di esecuzione t_n per calcolare lo stato fondamentale e l'energia totale per il sistema in oggetto. Il cut-off dell'energia è stato fissato a 1000 eV e si è considerata una griglia di punti k, 4x4x4, per lo spazio reciproco a cui corrispondono un totale di 36 punti k irriducibili. Quantum ESPRESSO permette di ripartire il carico di lavoro sulla base dei punti k utilizzando l'opzione *-npool(nk)*. Ad esempio, se scegliamo di utilizzare 12 nodi per l'esecuzione del calcolo e settiamo *-nk = 3*, allora, verranno creati 3 pool di 4 nodi ciascuno, e ciascun pool si prenderà carico di 12 punti k.
Di seguito riportiamo delle tabelle, una per ciascun valore di *-nk*, con i tempi di esecuzione t_n, lo speed-up *S* e l'efficienza *E*. Nella Figura 2 vengono riportati gli andamenti di queste grandezze al variare del numero di nodi core coinvolti nel calcolo.

Tabella 1. Risultati del benchmark per QE. Per ciascun valore del parametro -nk, si riporta il tempo di esecuzione del calcolo al variare del numero di nodi(core) utilizzati e i valori calcolati dello speed-up ed efficienza nella parallelizzazione.

-nk = 1

Nodi	1	2	3	4	5	6	7
Core	48	96	144	192	240	288	336
t (s)	15300	12240	10620	10320	9780	9780	10020
S	48	60.0	69.2	71.2	75.1	75.1	73.3
E (%)	100	62.5	48.0	37.1	31.3	26.1	21.8

-nk = 2

Nodi	1	2	3	4	5	6	7	8	9	10
Core	48	96	144	192	240	288	336	384	432	480
t (s)	14760	8340	7020	6660	5940	5640	5580	5580	5520	5340
S	49.8	88.1	104	110	124	130	132	132	133	138
E (%)	104	91.7	72.6	57.4	51.5	45.2	39.2	34.3	30.8	28.7

-nk = 3

Nodi	1	2	3	4	5	6	7	8	9	10
Core	48	96	144	192	240	288	336	384	432	480
t (s)	15660	7620	5820	4920	6000	4440	4380	3960	3840	3780
S	46.9	96.4	126	149	122	165	168	185	191	194

E (%)	97.7	100	87.6	77.7	51.0	57.4	49.9	48.3	44.3	40.5

-nk = 4

Nodi	1	2	3	4	5	6	7	8	9	10
Core	48	96	144	192	240	288	336	384	432	480
t (s)	16320	7980	5400	4260	3780	3600	3541	3370	3294	3121
S	45.0	92.0	136	172	194	204	207	218	223	235
E (%)	93.8	95.9	94.4	89.8	81.0	70.8	61.7	56.7	51.6	49.0

-nk = 6

Nodi	2	4	5	8	10	12	14	16	18	20	22	24
Core	96	192	288	384	480	576	672	768	864	960	1056	1152
t (s)	8040	3840	2923	2523	2405	2296	2248	2081	2046	1983	2037	2042
S	91.3	191	251	291	305	320	327	353	359	370	361	360
E (%)	95.2	99.6	87.2	75.8	63.6	55.5	48.6	46.0	41.6	38.6	34.1	31.2

-nk = 12

Nodi	2	4	5	8	10	12	14	16	18	20	22	24
Core	96	192	288	384	480	576	672	768	864	960	1056	1152
t (s)	9600	4140	2700	2024	1729	1538	1521	1334	1275	1236	1258	1251
S	76.5	177	272	363	425	477	483	550	576	594	584	587
E (%)	79.7	92.3	94.4	94.5	88.5	82.9	71.8	71.7	66.7	61.9	55.3	50.9

-nk = 18

Nodi	3	5	9	12	15	18	21	24	27	30
Core	144	288	432	576	720	864	1008	1152	1296	1440
t (s)	6300	2852	1829	1399	1180	1039	2097	911	896	872
S	117	257	402	525	622	707	350	806	819	842
E (%)	81.0	89.4	93.0	91.2	86.4	81.8	34.7	70.0	63.2	58.5

-nk = 36

Nodi	3	5	9	12	15	18	21	24	27	30	33	36
Core	144	288	432	576	720	864	1008	1152	1296	1440	1584	1728
t (s)	6960	3287	2007	1484	1133	955	856	739	692	668	621	574
S	106	223	366	495	648	769	858	994	1061	1100	1182	1279
E (%)	73.3	77.6	84.7	85.9	90.0	89.0	85.1	86.2	81.9	76.4	74.6	74.0

Ricordiamo che per il calcolo dello speed-up ed efficienza, abbiamo preso come riferimento il tempo di esecuzione del calcolo $t_{rif.}$ = 15300 s per il caso in cui si è utilizzato un solo nodo (48 core) e *-nk = 1*.

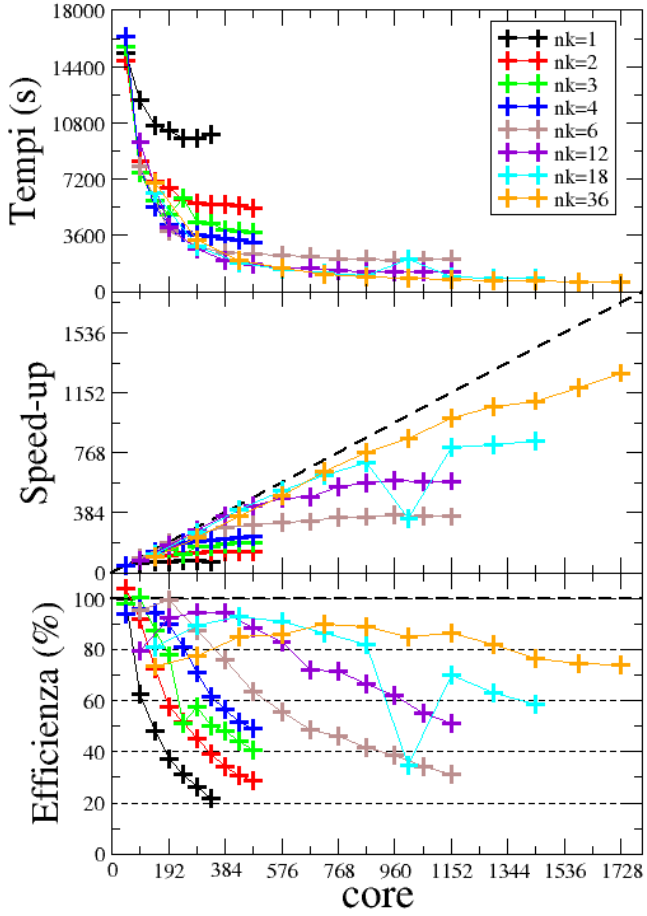

Figura 2. Risultati del benchmark per Quantum ESPRESSO. In alto, tempi di esecuzione del calcolo al variare del parametro *-nk*; al centro, speed-up del codice confrontato con quello ideale (linea tratteggiata nera); in basso, efficienza nella scalabilità parallela espressa in termini percentuali confrontata con quella ideale (linea tratteggiata nera).

Dall'analisi dei risultati si vede come, grazie alla possibilità di ripartire il carico tra i vari nodi di calcolo, con l'opzione *-nk = 36* si possa arrivare ad avere uno speed-up molto vicino a quello ideale fino a 36 nodi (1728 core) con un'efficienza nella scalabilità parallela vicino all' 80%. Infatti, passando da 1 nodo a 36 nodi di calcolo si riesce a ridurre il tempo di esecuzione da 15.300 s a 574 s con una riduzione di circa 27 volte.

3.2 Test di scalabilità di CP2K

In questo benchmark si analizza il tempo impiegato dal codice CP2K per calcolare lo stato fondamentale e l'energia totale del sistema composto da 1200 atomi e cella di simulazione di lato 2.4 nm. La formula chimica è $MAPbBr_3$ (Methylammonium Lead Tribromide) dove MA = CH_3NH_3 e la struttura atomica è rappresentata nella Figura 3.

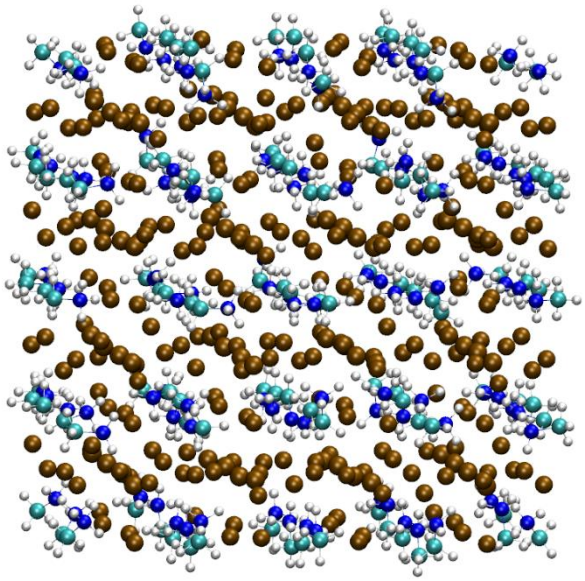

Figura 3. Modello atomistico di $MAPbBr_3$ (Methylammonium Lead Tribromide) dove MA = CH_3NH_3, che è il sistema di 1200 atomi utilizzato per il benchmark del codice CP2K.

Si tratta di una perovskite con promettenti capacità nel campo dei materiali per il fotovoltaico che rappresenta un altro campo di intervento per il progetto IEMAP (WP4).
Di seguito, nella Tabella 2, riportiamo i tempi impiegati dal codice per effettuare questo tipo di calcolo al variare del numero di nodi(core) utilizzati.

Tabella 2. Risultati del benchmark per CP2K. Si riporta il tempo di esecuzione del calcolo al variare del numero di nodi(core) utilizzati e i valori calcolati dello speed-up ed efficienza nella parallelizzazione.

Nodi	1	2	4	8	16
Core	48	96	192	384	768
Tempi esec. (s)	1020	692	438	320	255
Speed-up	48	70.8	112	153	192
Efficienza (%)	100	73.7	58.2	39.4	25

Questi valori sono riportati nella Figura 4, nella quale vengono anche mostrati gli andamenti dello speed-up e dell'efficienza nella scalabilità parallela confrontandoli con quelli ideali. Dall'analisi di queste grandezze si vede come si riesce ad avere una scalabilità sufficientemente accettabile solo fino a 192 core avendo un'efficienza al di poco sotto del 60%.
Nella prossima annualità di cercherà di incrementarla analizzando le varie opzioni di compilazione del codice e delle librerie matematiche richiamate. Inoltre, si valuteranno quali voci dell'input possono migliorare le performance del codice.

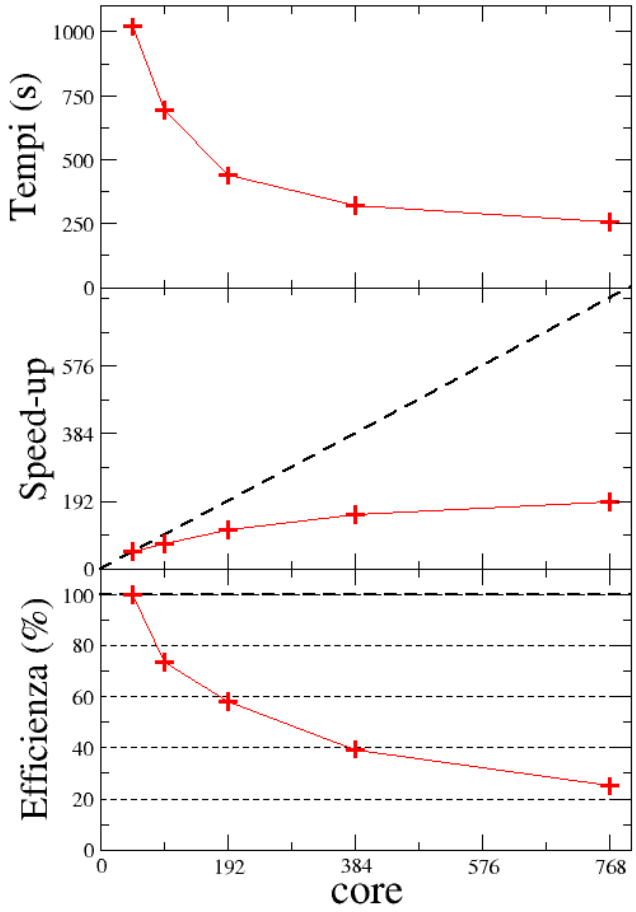

Figura 4. Risultati del benchmark per CP2K. In alto, tempi di esecuzione del calcolo; al centro, speed-up del codice confrontato con quello ideale (linea tratteggiata nera); in basso, efficienza nella scalabilità parallela espressa in termini percentuali confrontata con quella ideale (linea tratteggiata nera).

3.3 Test di scalabilità di VASP

Nel caso del codice VASP, sono stati considerati due sistemi di differenti dimensioni il primo, *test_case1*, di 216 atomi, il secondo, *test_case_2*, di 864 atomi.

Il primo sistema (*test_case_1*) è costituito di una etero-giunzione AlP/GaP (sistema studiato nel lavoro di Di Liberto e Pacchioni[1]), la cui cella di simulazione contiene 216 atomi. Il cut-off dell'energia è stato fissato a 405 eV e si è considerata una griglia di punti k, 4x4x1, per lo spazio reciproco. Nella tabella seguente (Tabella 3) sono riportati i tempi di esecuzione per 20 cicli SCF (Self Consistent Field) nel calcolo dello stato fondamentale ed energia totale.

[1] Di Liberto, G., & Pacchioni, G. (2021). Band offset in semiconductor heterojunctions. *Journal of Physics: Condensed Matter*, 33(41), 415002.

Tabella 3. Risultati del benchmark per VASP. Si riporta il tempo di esecuzione del calcolo per il *test_case_1* al variare del numero di nodi(core) utilizzati e i valori calcolati dello speed-up ed efficienza nella parallelizzazione.

Nodi	1	2	4	8	12	16
Core	48	96	192	384	576	768
T. esec. (s)	14289	7600	5105	4044	5281	8941
Speed-up	48	90.2	134	170	130	77
Effic. (%)	100	94.0	70.0	44.2	22.6	9.99

Il secondo sistema (*test_case_2*) si è ottenuto replicando per 4 quello precedente in modo da avere una cella di simulazione contenente 864 atomi. Il cut-off dell'energia è stato ridotto a 200 eV per evitare problemi con la memoria disponibile in ciascun nodo, e come prima, si è considerata una griglia di punti k, 4x4x1. Nella Tabella 4 sono riportati i tempi di esecuzione per 10 cicli SCF nel calcolo dello stato fondamentale ed energia totale. Per questo sistema si è preso come riferimento il tempo di calcolo relativo al caso in cui vendono utilizzati 2 nodi (96 core). Questa taglia di sistema non è eseguibile su un solo nodo a causa della richiesta di memoria superiore a quella disponibile su ciascun nodo che è pari a 192 GB.

Tabella 4. Risultati del benchmark per VASP. Si riporta il tempo di esecuzione del calcolo per il *test_case_2* al variare del numero di nodi(core) utilizzati e i valori calcolati dello speed-up ed efficienza nella parallelizzazione.

Nodi	2	4	8	12	16
Core	96	192	384	576	768
Tempi esec. (s)	21492	11784	6319	5577	6927
Speed-up	96	175	327	370	298
Efficienza (%)	100	91.2	85.0	64.2	38.8

La Figura 5 mostra i tempi di esecuzione del calcolo e i corrispondenti valori dello speed-up e dell'efficienza di parallelizzazione per i due sistemi. In rosso i dati relativi al primo sistema, in verde quelli relativi al secondo. Dall'analisi dei dati si vede che per il sistema piccolo la scalabilità parallela degrada dopo i 192 core dove è pari al 70%. Al contrario per il sistema replicato e quindi dimensionalmente più grande, si riesce ad avere un'efficienza di circa il 65% fino a 12 nodi.

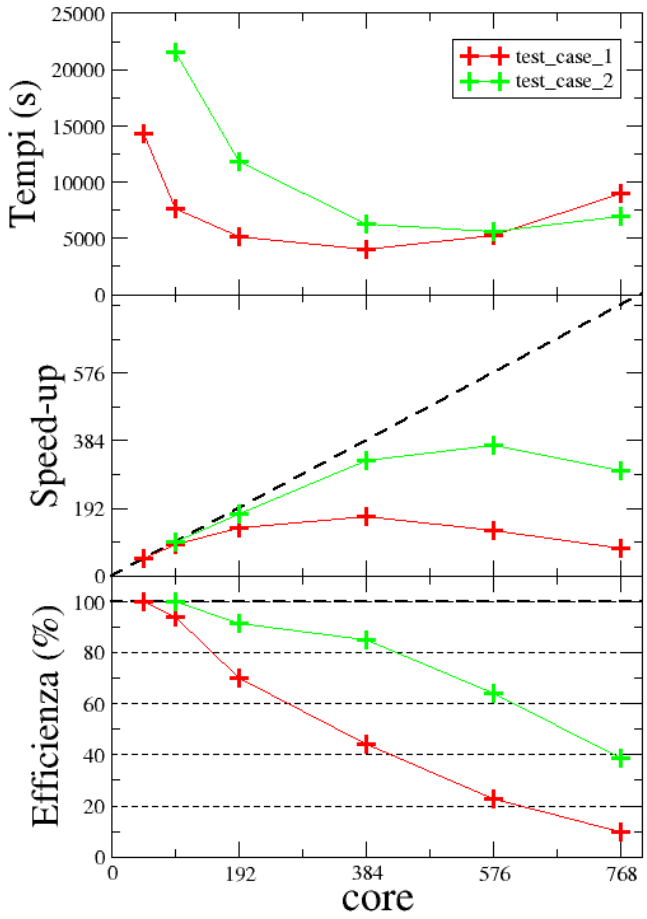

Figura 5. Risultati del benchmark per VASP. In alto, tempi di esecuzione del calcolo; al centro, speed-up del codice confrontato con quello ideale (linea tratteggiata nera); in basso, efficienza nella scalabilità parallela espressa in termini percentuali confrontata con quella ideale (linea tratteggiata nera).

4 Descrizione dei materiali oggetto di studio

In questa sezione riporteremo alcuni risultati preliminari sui materiali oggetto di studio nell'ambito del progetto IEMAP, per i quali sono stati utilizzati i codici precedentemente descritti.

4.1 Materiali catodici a base di sodio

Questa linea di attività si propone di studiare materiali catodici basati su ossidi metallici lamellari drogati e mediante un'approccio di calcolo *ab-initio* delle energie di formazione e dei potenziali di intercalazione. In particolare, l'attività è stata dedicata alla progettazione di nuovi materiali per catodi di batterie ricaricabili a base di ioni di sodio (NIB). L'obiettivo è quello di sostituire le diffuse batterie al litio (LIB), principalmente perché la limitata disponibilità di litio (Li) rende difficile soddisfare la prevista crescita della domanda di mercato. Al contrario, il sodio (Na) è ampiamente distribuito in tutto il mondo ed è poco costoso, rendendo le NIB un'alternativa promettente per le LIB. Inoltre, il sodio è più sicuro del litio che è un elemento raro ed è

molto reattivo all'ossigeno (le LIB possono esplodere). Sicurezza, buone prestazioni e costi inferiori, insieme al fatto che la stessa tecnologia e produzione già sviluppata per le LIB possono essere direttamente applicate alle NIB senza costi aggiuntivi, hanno suscitato un rinnovato interesse per questi sistemi.

Gli svantaggi, al momento, riguardano la densità di energia e le prestazioni del ciclo: nelle NIB la densità di energia è ancora inferiore rispetto alle LIB e la struttura geometrica può subire una distorsione Jahn Teller al momento della scarica/ricarica. Molti studi sono dedicati all'aumento della densità di energia delle NIB, studiando come migliorarne le prestazioni attraverso il miglioramento dei catodi. Vari studi hanno mostrato che il drogaggio è una tecnica promettente per limitare questi svantaggi.

Partendo dal materiale $NaMnO_2$ e dopandolo attraverso la sostituzione di atomi di Mn con metalli di transizione come Ti e Ni si è visto che ciò migliora la capacità di ciclabilità e la densità di energia. In particolare, ciò si realizza quando si hanno concentrazioni simili di questi tre elementi e la loro disposizione spaziale è la più uniforme possibile. Nella Figura 6, sono mostrate rispettivamente le configurazioni atomiche del materiale $NaMnO_2$ (in alto a sinistra) al quale è stato sostituito un atomo di manganese con uno di nichel (in alto a destra) e con uno di titanio (in basso a sinistra). Infine, l'ultima configurazione (in basso a destra) si riferisce ad una di quelle con migliori proprietà in termini di energia di formazione e potenziale di intercalazione. In questa particolare configurazione, si hanno 5 atomi di Mn, 4 di Ni e tre di Ti. Questi atomi sono distribuiti sui due *layer* $L_{up,down}$ in modo da avere su L_{up}: 2 Mn + 2 Ni + 2 Ti: e su L_{down}: 3 Mn + 2 Ni + 1 Ti.

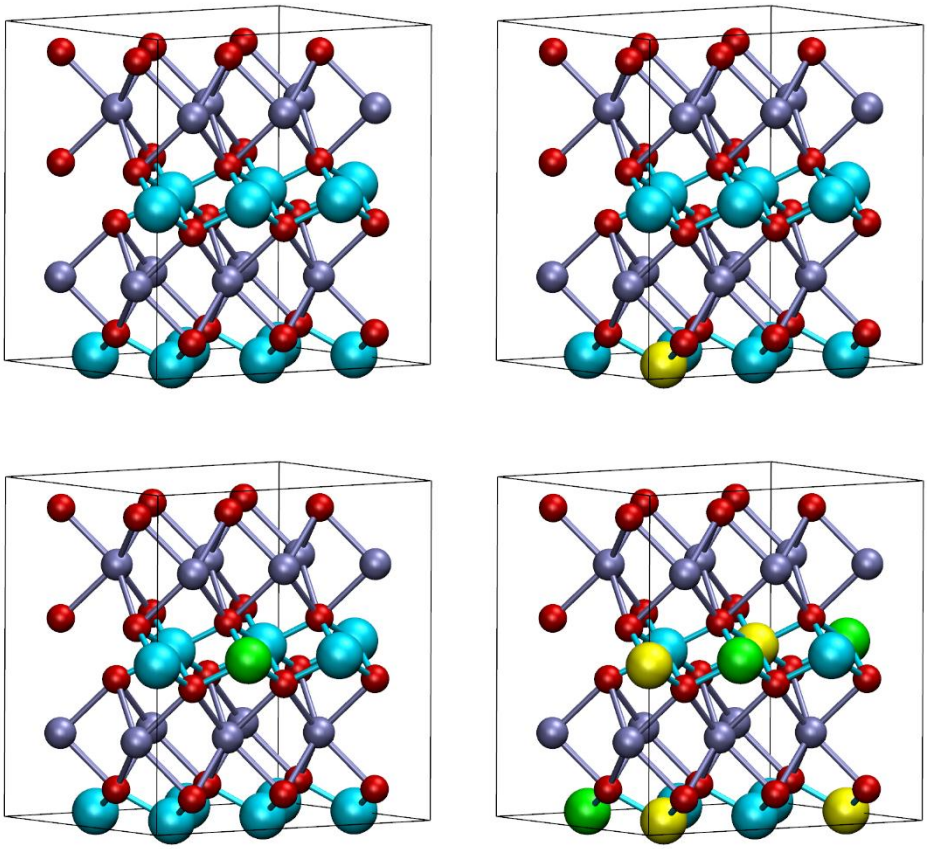

Figura 6. Modello atomistico di $NaMnO_2$ e varianti a seguito del dopaggio sostitutivo del Mn con Ni e Ti.

4.2 *Perovskiti per il fotovoltaico*

I sistemi investigati sono nanocristalli (NCs) di doppie perovskiti (DPs), una classe di semiconduttori caratterizzati da un reticolo tridimensionale di ottaedri $[B^{+}X_6]$ e $[B^{3+}X_6]$ connessi per vertici con i voluminosi cationi A che riempiono gli spazi intermedi, presentando una stechiometria complessiva $A_2B^{+}B^{3+}X_6$. Nel particolare, sono stati considerati NCs di DPs del tipo $Cs_2Ag_{1-x-y}Na_xK_yInCl_6$ dopate con Bi^{3+}. Nella figura seguente viene mostrato uno di questi sistemi.

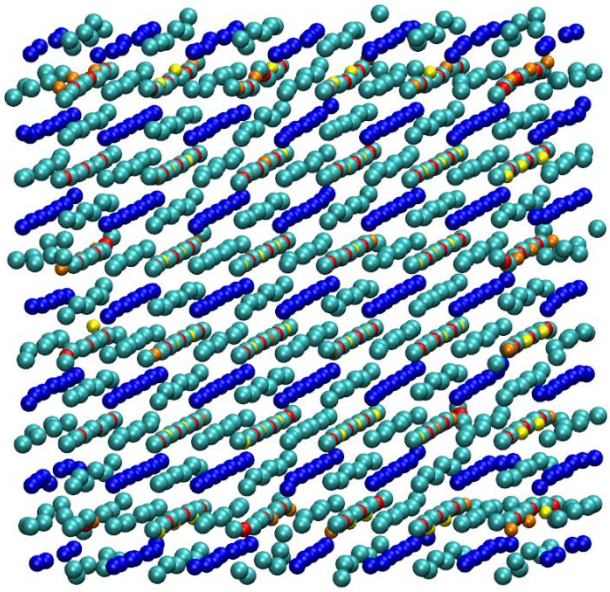

Figura 7. Modello atomistico del nanocristallo di doppie perovskiti del tipo $Cs_2Ag_{1-x-y}Na_xK_yInCl_6$ dopate con Bi^{3+}.

Per rivelare la struttura atomistica ed elettronica di questi complessi sistemi, abbiamo utilizzato calcoli quanto-meccanici, ed in particolare teoria del funzionale della densità utilizzando il codice CP2K.
In primo luogo, abbiamo studiato come la modulazione della composizione ionica dei NCs di DPs determina il loro meccanismo di ricombinazione radiativa: per eliminare i complicati effetti di superficie, sono stati utilizzati i corrispondenti sistemi bulk come modelli computazionali. I nostri calcoli di struttura elettronica allo stato fondamentale ed eccitato hanno evidenziato come l'introduzione di uno ione sostituente (in concentrazioni di drogaggio o di lega) nel sistema ospite di DP possa portare all'emergere di stati localizzati, la cui energia e la sovrapposizione dipende anche dal tipo di matrice DP circostante, determinando in ultima battuta la formazione di efficienti percorsi di ricombinazione radiativa.
È inoltre noto come un'incompleta passivazione superficiale dei NC colloidali, che risulta nella presenza di ioni sotto-coordinati sulla loro superficie, possa portare all'emergere di stati elettronici di mid-gap (i cosiddetti "stati trappola") e aprire efficienti percorsi non radiativi. In secondo luogo, sono quindi stati costruiti modelli atomistici di NCs di DPs che tengono esplicitamente conto di questi effetti di superficie. I nostri calcoli DFT hanno indicato che anche una piccola frazione di siti superficiali non passivati è sufficiente a creare stati trappola nei NCs di DPs, contrariamente al caso delle NC di perovskite a base di Pb che presentano una tolleranza ai difetti superficiali molto più elevata.

4.3 Materiali per dispositivi optoelettronici

I nanocristalli (NCs) semiconduttori, detti anche *quantum dots* (QD), offrono il vantaggio di avere proprietà optoelettroniche che non solo si differenziano da quelle dei materiali macroscopici, ma sono anche regolabili in base alla necessità del loro impiego tecnologico. I "*core-shell*" QD sono formati da un nanocristallo di un dato materiale (core) sulla cui superficie viene cresciuto uno strato di un secondo materiale (shell). Questi tipi di QD offrono, tra le altre proprietà, delle migliori prestazioni in termini di resa quantistica di fotoluminescenza.

Nel nostro lavoro abbiamo utilizzato simulazioni di chimica quantistica per studiare i QD formati da un core di InAs e una shell di ZnSe. Vista la complessità del sistema, che rende impossibile determinare la struttura del QD da principi primi (si veda la discussione nella sezione S7 del lavoro[2]) abbiamo utilizzato un approccio "trial and error": abbiamo sistematicamente variato la composizione atomistica del nanocristallo sino a trovarne uno la cui composizione elementale e la dimensione fossero in accordo coi dati sperimentali, e che mostrasse una struttura elettronica compatibile con gli spettri di luminescenza misurati sperimentalmente. Questa procedura innovativa ci ha permesso di scoprire quali caratteristiche strutturali conferiscono ai nanocristalli sintetizzati dai nostri colleghi sperimentali le eccellenti proprietà ottiche, ad esempio la presenza di un layer intermedio tra core e shell formato da In, Zn, As e vacanze cationiche.

Le simulazioni sono state effettuate col software VASP, principalmente su cresco6, utilizzando tipicamente 526 o 1052 core (rispettivamente per la struttura elettronica e per la più computazionalmente impegnativa ottimizzazione di geometria).

In Figura 8, viene mostrato un QD formato da un core di InAs e una shell di ZnSe.

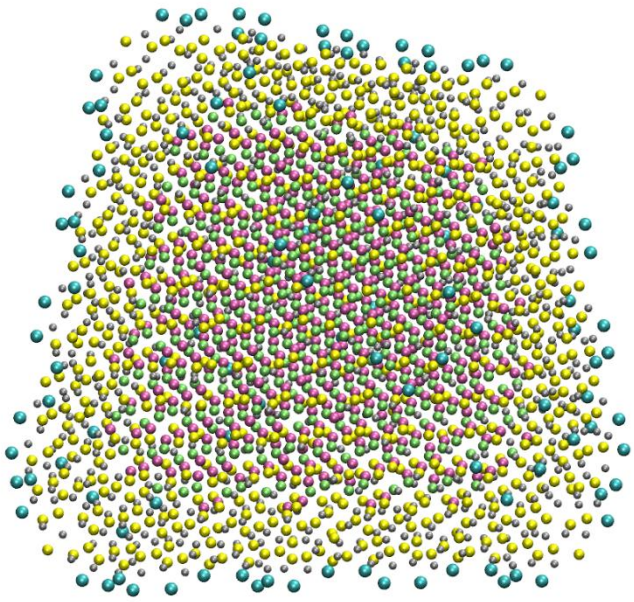

Figura 8. Modello atomistico del *quantum dots* formato da un core di *InAs* e una shell di *ZnSe*.

[2] Zhu, D., *et al.* (2023). Boosting the Photoluminescence Efficiency of InAs Nanocrystals Synthesized with Aminoarsine via a ZnSe Thick‐shell Overgrowth. *Advanced Materials*, 2303621.

5 Codici di calcolo utilizzando tecniche di deep learning

L'Intelligenza Artificiale (IA) rappresenta un elemento cruciale nella Scienza dei Materiali, consentendo previsioni più accurate e una comprensione più profonda dei materiali cristallini tramite l'analisi accurata 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. I dati possono essere trovati anche i database esterni che vengono citati nelle sezioni successive.

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 Jiucheng, Cheng, Dong, Lifeng 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).

Sulla base di questa libreria, è stato prodotto il codice riportato sul repository GitLab ENEA al link: https://gitlab.brindisi.enea.it/claudio.ronchetti/ai4mat.

5.1 Ambiente virtuale

Sono state progettate diverse versioni di contenitori Singularity, ognuna delle quali supporta diverse reti neurali basate su deep learning e basate su PyTorch nella scienza dei materiali.

E' possibile scaricare le immagini software di Singularity dal catalogo "ai4mat" della pagina web CrescoWare (https://crescoware.brindisi.enea.it/).

Le versioni al momento disponibili sono:

- Supporto per CUDA 10.1
  - *Versione di PyTorch 1.6.0*
    - 1.6.0-cuda10.1-devel, container PyTorch (v.1.6.0) compilato con la versione CUDA 10.1 per macchine con la stessa versione CUDA.
  - *Versione di PyTorch 1.7.1*
    - alignn-gdl-cu101, questo container è stato creato per eseguire reti neurali basate su deep learning chiamate Atomistic Line Graph Neural Network (ALIGNN). ALIGNN introduce uno strato di convoluzione grafica che modella esplicitamente le interazioni a due e tre corpi nei sistemi atomici.
- Supporto per CUDA 11.3
  - *Versione di PyTorch 1.10.0*
    - 1.10.0-cuda11.3-devel, container PyTorch (v.1.10.0) compilato con la versione CUDA 11.3 per macchine con la stessa versione CUDA.
    - alignn-gdl-cu111, questo container supporta Atomistic Line Graph Neural Network (ALIGNN) con la versione CUDA 11.3.

Innanzitutto, per sviluppare e utilizzare la rete neurale artificiale (ANN), è necessario creare un ambiente che installi le librerie di machine learning Python (PyTorch) e altre utility.
Il nucleo dell'ambiente è la libreria PyTorch compilata per sfruttare la GPU. Pertanto, ci sono due modi principali per raggiungere questo obiettivo:

1. Partendo da una distribuzione Ubuntu, è necessario includere tutti i pacchetti da installare nella ricetta di Singularity (vedi Singularity-tutorial).
2. Oppure, il modo più semplice, partendo da un container Docker pre-builded da PyTorch Corp.

Seguendo il secondo metodo, iniziamo cercando un container Docker pre-builded archiviato in Official-Pytorch-Docker-Hub.
Successivamente, è necessario conoscere la versione di CUDA installata sulla computer (con GPU) utilizzando il comando 'nvidia-smi'.

Su Docker Hub, è possibile selezionare un TAG di PyTorch con la stessa o una versione precedente di CUDA.

I comandi utilizzati per la costruzione della versione *1.6.0-cuda10.1-devel* è la seguente:

```
Bootstrap: docker
From: pytorch/pytorch:1.6.0-cuda10.1-cudnn7-devel
IncludeCmd: yes
%post
  PATH=/opt/conda/bin:$PATH
  pip install pymatgen boto3
```

Se il computer con GPU ha una versione di CUDA diversa, è necessario modificare il tag nella sezione 'From' (seconda riga della ricetta) con il tag corretto.

Questo container consente di eseguire script Python che richiedono la libreria PyTorch. E' possibile eseguire lo script con il seguente comando:

```
singularity exec --nv ai4mat-pytorch.simg \
   python path/to/script.py --opt1 valore1 --opt2 valore2 ...
```

NOTA: L'opzione --nv è necessaria per eseguire i calcoli Python sfruttando la potenza della GPU.

5.2 Materials Project Data Extractor

Tuttavia, la chiave del successo nell'applicazione dell'IA in questo campo è la disponibilità di dati di alta qualità. Senza dati adeguati non è possibile sfruttare una serie di tecniche di Deep Learning (DL). In questi casi, è possibile considerare l'utilizzo di database esterni. Alcuni dei principali database mondiali di strutture cristalline includono AFLOW, COD, TCOD, Materials Project, Materials Cloud NOMAD, ODBX, Open Materials Database (OMDB) e OQMD.

A tal proposito, è stato sviluppato un tool chiamato Materials Project Data Extractor (MP) per IEMAP. Questo strumento utilizza direttamente l'API REST di MP per scaricare i dati sui materiali cristallini dal database. Questo strumento è risultato estremamente utile per l'addestramento dei modelli di apprendimento automatico e per la previsione di proprietà dei materiali, come ad esempio l'energia di formazione.

Tale codice è uno script Python che estrae dati dal Materials Project Database (MP) e li salva come file CIF nella directory specificata.

Per eseguire lo script, è necessario utilizzare il comando:

```
python mp_data.py
```

Il processo di download richiede di salvare temporaneamente i dati di MP in un file JSON predefinito (data.json) o in un file JSON personalizzato (usando l'opzione --json-data) con una dimensione di circa 250 MB. Se i dati sono stati già caricati e non si desidera ripetere il passaggio di download, è possibile utilizzare l'opzione --no-download.

Inoltre, l'utente può applicare un filtro per scaricare un sottoinsieme dei dati di MP modificando il file JSON filter.json. Se il file per il filtro è vuoto, nessun criterio viene applicato.

Ad esempio, è possibile usare la chiave task_id per scaricare un materiale specifico utilizzando l'ID del task, che in realtà è l'ID dei materiali nel Materials Project:

```
{"task_id": "mp-1234"}
```

Oppure è possibile cercare il materiale utilizzando la chiave pretty_formula:

```
{"pretty_formula": "Li2O"}
```

Alcuni filtri interessanti includono la selezione di strutture con più di o uguale a due elementi unici:

```
{"nelements":{"$gte": 2}}
```

La selezione di strutture che contengono elementi unici nella lista ["Na", "Mn", "O"]:

```
{"elements":{"$in" : ["Na", "O", "Mn"]}}
```

O la selezione di strutture che non contengono elementi unici nella lista ["He", "Ne", "Ar", "Kr", "Xe"]:

```
{"elements":{"$nin" : ["He", "Ne", "Ar", "Kr", "Xe"]}}
```

Per ulteriori dettagli, è possibile consultare la pagina relativa alle API di Materials Project.

L'ultima fase del codice genera tre statistiche per mostrare all'utente le principali caratteristiche del dataset scaricato e le salva nella directory ml/data/stats. Le statistiche includono:

- *Conteggio delle strutture in base al numero di elementi* – Riportata in Figura 1, la prima statistica mostra che la maggior parte delle strutture ha da due a cinque elementi unici.

- *Conteggio degli elementi nelle strutture, visualizzato tramite la tavola periodica* – Riportata in Figura 2, la seconda statistica fa notare che l'elemento più presente nel dataset di MP è l'ossigeno (O), con oltre 60.000 strutture, mentre gli elementi meno presenti sono l'attinio (Ac) e il protattinio (Pa) con circa 300 strutture ciascuno. Gli elementi colorati in grigio non sono presenti nel dataset di MP.

- *Energia di formazione media per ciascun elemento* – Riportata in Figura 3, la terza statistica mostra che l'elemento con l'energia di formazione media più alta è il tecnizio (Tc), con oltre 0,5 eV/atomo, mentre l'elemento con l'energia di formazione media più bassa è il fluoro (F), con meno di -2,5 eV/atomo. Gli elementi colorati in grigio non sono presenti nel dataset di MP. Alcuni dei metalli di transizione più interessanti sono il vanadio (V) e il titanio (Ti), entrambi con meno di -2,0 eV/atomo.

È possibile saltare questa fase utilizzando l'opzione --no-stats.

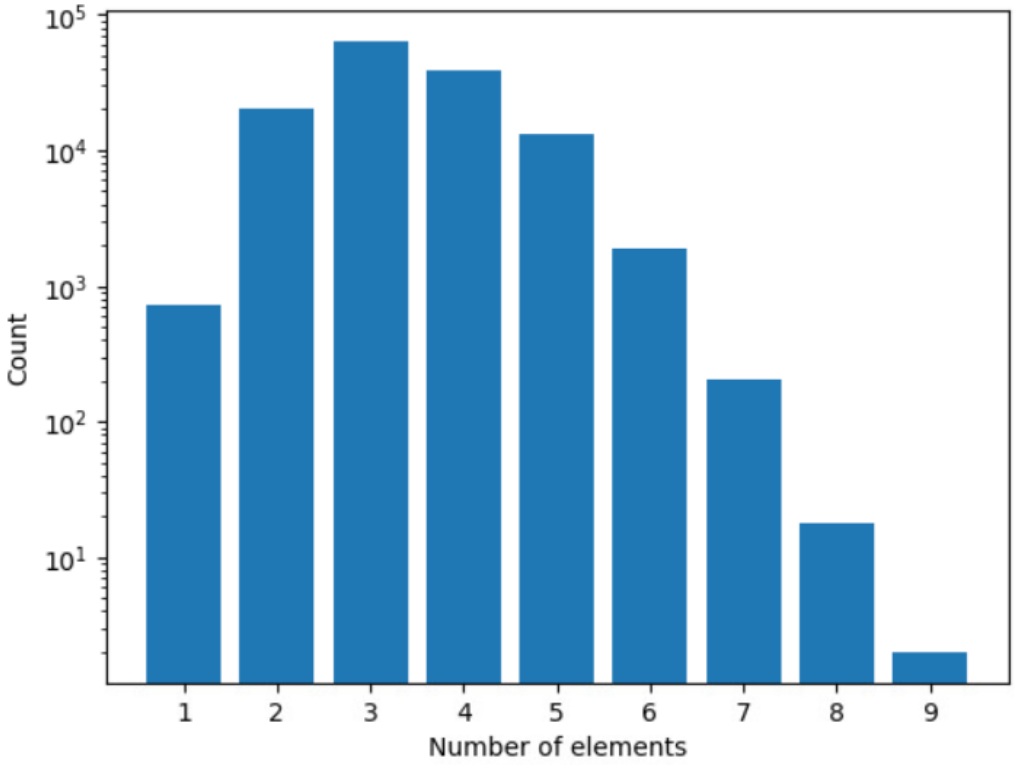

Figura 1. Conteggio delle strutture in base al numero di elementi

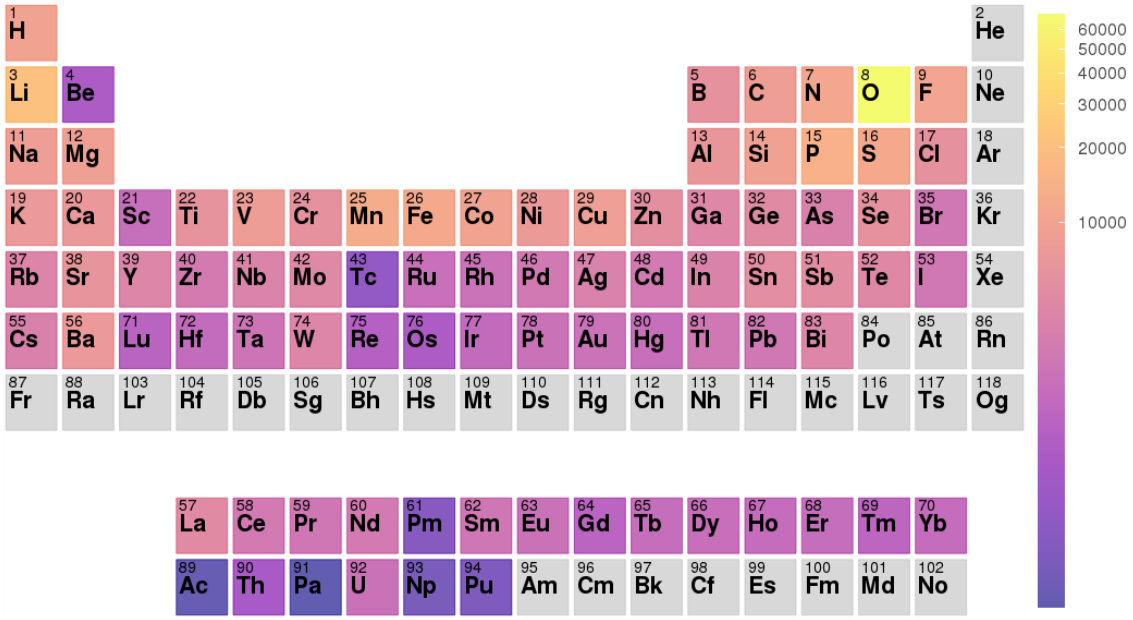

Figura 2. Conteggio degli elementi nelle strutture, visualizzato tramite la tavola periodica

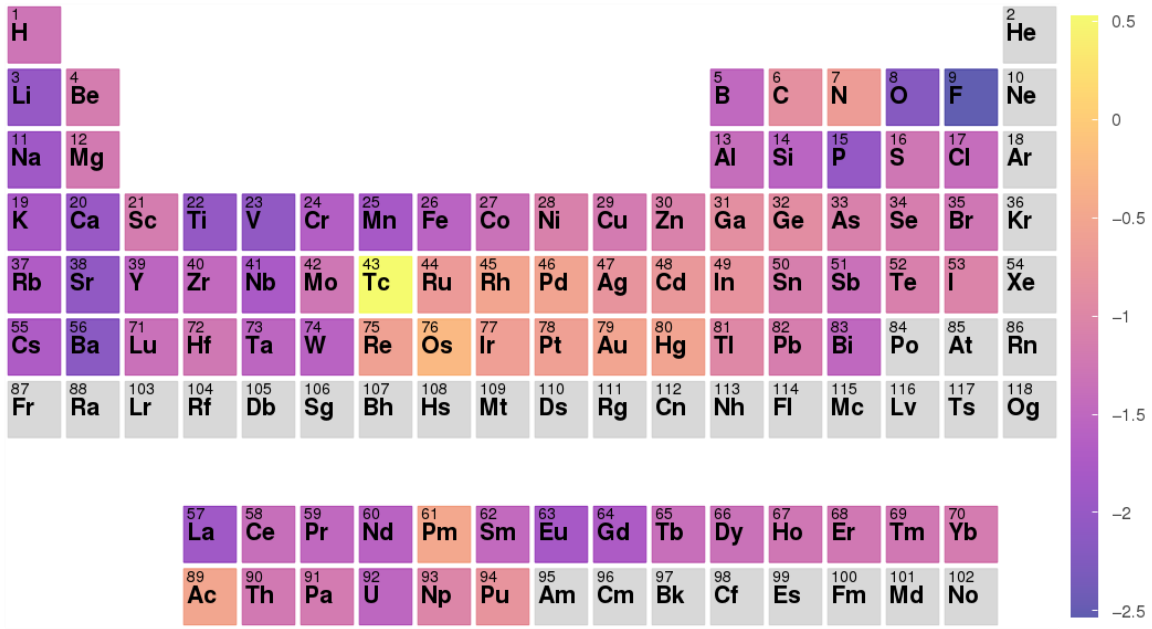

Figura 3. Energia di formazione media per ciascun elemento

5.3 Quick start

Per iniziare rapidamente a utilizzare GeoCGNN, è possibile seguire i seguenti passaggi:

1. Assicurarsi di utilizzare un nodo CRESCO che supporti GPU tramite SSH.
2. Scaricare il dataset di Materials Project (CIF, energia di formazione) utilizzando il seguente comando:

```
cd <path-to>/ai4mat/ml/data-extractor/
DATASET=<path-to-download>
python mp_data.py $DATASET
```

3. Avviare l'addestramento utilizzando il dataset di Materials Project:

bashCopy code

```
# cambia la directory di lavoro
cd <percorso-a>/ai4mat/ml/geo-gcnn/
# carica l'ambiente Singularity module load singularity-3.5.2
SIMG= path-to/ai4mat-pytorch_1.6.0-cuda10.1-devel.simg
 # avviare l'addestramento
singularity exec -B /gporq2 --nv $SIMG \
   python main.py --clean-torch --storage file --batch-size 64 --lr 0.001 \
   --optim Adam --train-ratio 0.7 --val-ratio 0.2 --test-ratio 0.1 --workers 20 \
   --epochs 300 --print-freq 1 $DATASET
```

NOTA: Si consiglia di utilizzare una macchina con supporto GPU.

4. Eseguire l'inferenza sui dati di esempio:

```
# cambia la directory di lavoro
cd <percorso-a>/ai4mat/ml/geo-gcnn/
# copia il file di configurazione nella directory del dataset cp
PERCORSO_DATASET/config_onehot.json ../../example/data
# carica l'ambiente Singularity module load singularity-3.5.2
SIMG=/gporq3/store_0/usr/aiidausr/SINGULARITY/containers/ai4mat-pytorch_1.6.0-cuda10.1-devel.simg
# avviare l'inferenza
singularity exec --nv $SIMG python predict.py --modelpath output/model_best.pth.tar --workers 10 --print-freq 1 ../../example/data
```

Le previsioni saranno salvate in **<path-to>/ai4mat/ml/geo-gcnn/output_pred**.
Il processo di apprendimento (addestramento) di una rete neurale è un processo iterativo in cui i calcoli vengono eseguiti in avanti e indietro attraverso ogni strato della rete fino a quando la funzione obiettivo è minimizzata. GeoCGNN richiede tre tipi di file nella stessa directory per l'addestramento:

1. **CIF**: tutti i file CIF che registrano le strutture cristalline nel formato **<ID>.cif**, dove ID è l'identificativo univoco del cristallo.
2. **Label**: un file CSV con due colonne. La prima colonna contiene un ID univoco per ogni cristallo, mentre la seconda colonna contiene il valore della proprietà target.
3. **Config**: un file JSON che registra il numero di elementi atomici utilizzando lo standard onehot.

La prassi migliore è salvare tutti i dati nell'area di archiviazione GPFS come **/gporq2/scratch_0/usr/<tuo-utente>/ml_dataset** o nel bucket CEPH chiamato *ai4mat*.
Puoi seguire questa guida su come scaricare il dataset di Materials Project.
Ecco un esempio di comando per avviare l'addestramento di GeoCGNN:

```
singularity exec -B /gporq2 --nv $SIMG python main.py \
  --storage file --batch-size 32 --lr 0.001 --optim SGD \
  --train-ratio 0.7 --val-ratio 0.2 --test-ratio 0.1 \
  --workers 20 --epochs 100 --print-freq 1 /
  <path/to/dataset>
```

NOTA: Si consiglia l'utilizzo di una macchina con supporto GPU per accelerare il processo di addestramento.

Come Eseguire il Test
Per eseguire il test di GeoCGNN, è necessario caricare i file CIF, di configurazione e di etichetta nella stessa directory. Se si desidera prevedere solo la proprietà dei materiali, è possibile creare un file di etichetta in cui la prima colonna contenga l'ID CIF e la seconda colonna (la vera proprietà del materiale) sia 0 (zero).
Ecco un esempio di comando per eseguire il test:

```
singularity exec -B /gporq2 --nv $SIMG python predict.py \
  --print-freq 1 \
  --modelpath experiments/models/AI4MATXOR.pth.tar \
  --workers 10 \
  <path/to/dataset>
```

Le previsioni saranno salvate in **<percorso-a>/ai4mat/ml/geo-gcnn/output_pred/prediction.csv**. Il file CSV di output contiene tre colonne: l'ID CIF, la vera proprietà del materiale (0 se non disponibile) e la proprietà del materiale prevista.

6 Conclusioni

In questo documento abbiamo riassunto l'attività svolta nel secondo anno di finanziamento del progetto IEMAP riguardante la LA1.2. L'attività si è articolata in modo tale da: fornire la formazione agli utenti della piattaforma IEMAP per l'utilizzo ottimale di CRESCO e dei servizi ENEAGRID; pianificare le attività per ciascun utente sull'infrastruttura di supercalcolo CRESCO; definire un ambiente dedicato per l'installazione di tutti i codici di calcolo necessari per l progetto IEMAP; realizzare dei benchmark di questi codici di calcolo. Inoltre è stata creato un ambiente di utilizzo di librerie machine learning da utilizzare per l'analisi dei materiali.

7 Abbreviazioni ed acronimi

HPC = High-Performance Computing
AFS = Andrew File System
ACL = Acces Control List
GPFS = General Parallel File System
DFT = Density Functional Theory
SCF = Self Consistent Field
GNU = GNU's Not Unix
GPL = General Public License
MPI = Message Passing Interface
OpenMP = Open Multi Processing)
QE = Quantum ESPRESSO
VASP = Vienna Ab-initio Simulation Package
LAMMPS = Large-scale Atomic/Molecular Massively Parallel Simulator
GPU = Graphics Processing Unit
NIB = Na-Ion Battery
LIB = Li-Ion Battery
NC = Nano Cristalli
DP = Doppie Perovskiti
QD = Quantum Dots

Bibliografia

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

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 Volume 177, Issues 1–2*.

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* .