Title

The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on REpresentational State Transfer (REST) principles

Authors

Ong, Shyue Ping
Cholia, Shreyas
Jain, Anubhav
et al.

Publication Date

2015-02-01

DOI

10.1016/j.commatsci.2014.10.037

Peer reviewed

Computational Materials Science 97 (2015) 209–215

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

The Materials Application Programming Interface (API): A simple, flexible and efficient API for materials data based on REpresentational State Transfer (REST) principles

Shyue Ping Ong [a,*], Shreyas Cholia [b], Anubhav Jain [b], Miriam Brafman [b], Dan Gunter [b], Gerbrand Ceder [c], Kristin A. Persson [b]

[a] Department of NanoEngineering, University of California, San Diego, 9500 Gilman Drive, Mail Code 0448, La Jolla, CA 92093, USA
[b] Lawrence Berkeley National Lab, 1 Cyclotron Rd, Berkeley, CA 94720, USA
[c] Department of Materials Science and Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA

ARTICLE INFO

Article history:
Received 18 August 2014
Accepted 18 October 2014

Keywords:
Materials Project
Application Programming Interface
High-throughput
Materials genome
Rest
Representational state transfer

ABSTRACT

In this paper, we describe the Materials Application Programming Interface (API), a simple, flexible and efficient interface to programmatically query and interact with the Materials Project database based on the REpresentational State Transfer (REST) pattern for the web. Since its creation in Aug 2012, the Materials API has been the Materials Project's *de facto* platform for data access, supporting not only the Materials Project's many collaborative efforts but also enabling new applications and analyses. We will highlight some of these analyses enabled by the Materials API, particularly those requiring consolidation of data on a large number of materials, such as data mining of structural and property trends, and generation of phase diagrams. We will conclude with a discussion of the role of the API in building a community that is developing novel applications and analyses based on Materials Project data.

1. Introduction

First principles methods are today a critical tool in the study and design of materials. Starting from the fundamental laws of physics with minimal assumptions and approximations, first principles techniques can access a wide range of chemistries in a relatively agnostic manner, making them especially powerful in materials investigations or design problems spanning diverse chemical spaces.

In the past decade, electronic structure calculation codes [1–4] have reached a level of maturity that it is now possible to reliably automate and scale first principles calculations across any number of compounds. Coupled with computing advances, this development has led to the advent of high throughput (HT) first principles calculations as an investigative and design tool in materials science. Even today, there are already several examples of HT first principles computation-guided materials design efforts in applications as varied as alkali-ion batteries [5–9], catalysts for hydrogen production [10], topological insulators [11], and organic semiconductors [12], with many of these efforts resulting in the discovery of novel materials that have already been synthesized and verified experimentally. This HT capability has also spurred the development of large databases of computed data on materials, such as the Materials Project [13], the AFLOWLIB library [14] and the Harvard Clean Energy Project [12].

In particular, the Materials Project [13], created by the authors of this paper, has led the charge of combining a large database of materials properties with a diverse and growing set of online analysis and comprehensive open source software tools [15–17]. The Materials Project's database today contains computed energetic properties for over 59,000 crystal structures along with over 25,000 electronic structure properties. More structures and properties (*e.g.*, elastic constants, dielectric constants, etc.) are being added on a daily basis. A series of web applications provide users with the capability to perform advanced searches and common analyses such as phase diagram and Pourbaix diagram generation [18–20], reaction energy computations, prediction of novel structures [21,22], etc. However, while these web applications provide user-friendly graphical interfaces to explore materials data and analyses, they do not provide easy programmatic access to the underlying resources or a means for the community to develop novel applications or analyses.

* Corresponding author.
E-mail addresses: ongsp@ucsd.edu (S.P. Ong), scholia@lbl.gov (S. Cholia), ajain@lbl.gov (A. Jain), mbrafman@lbl.gov (M. Brafman), dkgunter@lbl.gov (D. Gunter), gceder@mit.edu (G. Ceder), kapersson@lbl.gov (K.A. Persson).
URLs: http://www.materialsvirtuallab.org (S.P. Ong), http://ceder.mit.edu (G. Ceder).

http://dx.doi.org/10.1016/j.commatsci.2014.10.037
0927-0256/

In this paper, we describe the Materials Application Programming Interface (API), a simple, flexible and efficient interface to programmatically query and interact with the Materials Project database based on REpresentational State Transfer (REST) principles [23]. The provision of RESTful web services as a complementary method of accessing online resources is not a new concept. Most notably, the Protein Data Bank (PDB), which shares similar aims as the Materials Project but in a different scientific domain, has implemented such web services for a number of years [24]. However, the solid-state community has generally lacked the adoption of such information exchange protocols, and to our knowledge, the Materials API is the first API of its kind for solid-state materials data. The Materials Project has also implemented a high-level interface to the Materials API in the open-source Python Materials Genomics (pymatgen) materials library [15], which provides a reference implementation for accessing the API and supporting analysis tools. Since its creation in Aug 2012, the Materials API has been the Materials Project's *de facto* platform for data access, supporting not only the Materials Project's many collaborative efforts but also enabling new applications and analyses. This paper will highlight several examples of these applications and analyses. With increasing demand for federated materials data storage and sharing, we hope that the Materials API can also inspire other materials data collections (*e.g.*, the AFLOWLIB consortium) [25] to adopt similar flexible, high throughput data interfaces based on open standards.

2. Overview of the Materials Project database

Before we embark on a discussion of the Materials API and its design, it is necessary to first provide a brief overview of the underlying Materials Project database structure from which the data requests are served. The vast majority of data currently available through the Materials Project is *computed first principles data*. This data is computed using tens of millions of CPU-hours at the National Energy Scientific Computing Center (NERSC). The type of data generated falls into three broad categories (see Fig. 1):

1. Information about the individual computing **tasks**, including input parameters, raw calculation output, history (*e.g.*, originating structures, error correction protocols applied, etc.) and file locations. As of this writing, over 240,000 successful tasks have been completed.
2. Consolidated information about individual **materials**. Several electronic structure calculations (or tasks) with different "task types" are generally performed in order to compute multiple properties of a single material (*e.g.*, structure, energy, band structure, elastic tensor, etc.). At this level, the details of the individual tasks are generally removed in favor of physical properties revealed by the calculations. Currently, data is available for over 55,000 materials.
3. Higher level **analysis** data (often tailored to applications), which can combine information from several materials. For example, a battery electrode combines information from at least two materials at different states of charge. A phase diagram combines energy data from all materials in a chemical space.

All data are stored in MongoDB, a NoSQL database in which each individual task, material, or analysis-specific entity is stored as a single BSON document. For example, the **materials** collection contains over 55,000 such documents, one for each material. Each document has a corresponding identifier or id (*e.g.*, task_id, materials_id, or battery_id) that can be used to uniquely identify a calculation, material, or battery. For large data (*e.g.*, band structures), we use GridFS collections for storage. Each category of data (tasks, materials, or analysis data) is stored in a MongoDB collection, which corresponds roughly to a table in traditional SQL databases. This organization mirrors the fact that each type of data has a different document structure.

2.1. Data generation

We provide herein a brief overview of the process by which the data in the Materials Project database is generated (Fig. 2).

The calculation process begins when a user or algorithm submits a crystal structure to a submissions database. The crystal structure might originate from a structural database such as the Inorganic Crystal Structure Database (ICSD) [26] or it might be a proposed new compound. The distinction is recorded within the StructureNL object in the *pymatgen* codebase [15], which tracks the history of each crystal structure.

Once the compound is submitted, the rest of the calculation process is fully automated. The submission is detected by a background process and automatically mapped to a calculation

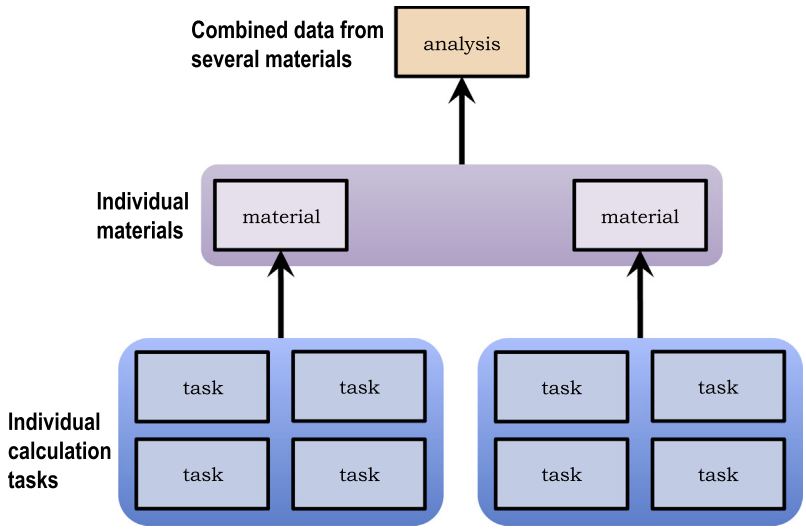


Fig. 1. Different levels of data in Materials Project database: tasks, materials, and analysis data such as battery electrodes.

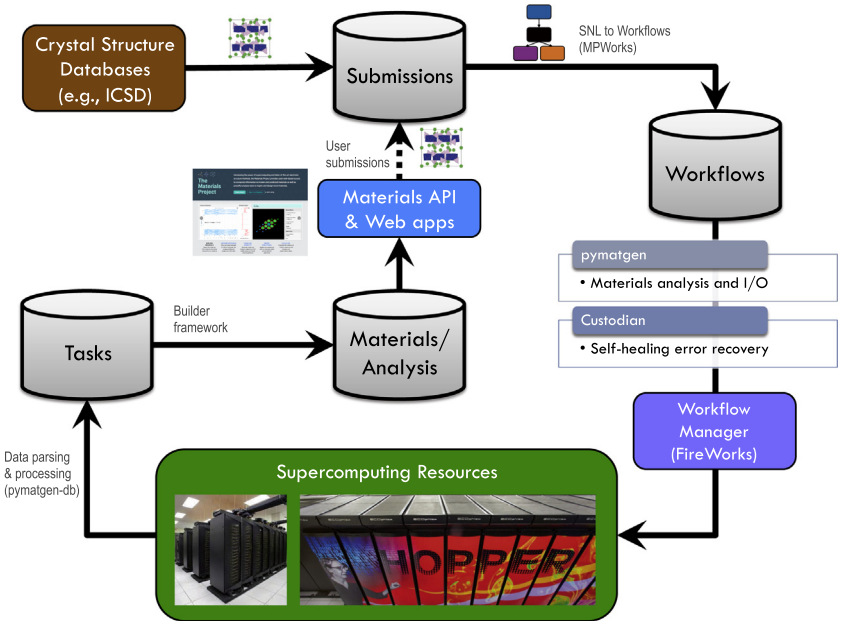


Fig. 2. The Materials Project computation infrastructure.

workflow that specifies task types (*e.g.*, structure, energy, band structure) and their dependencies. The workflows are stored and executed at NERSC using the *FireWorks* code [17], which also automatically checks for duplicated jobs. Internally, the jobs use *pymatgen* to set up the input files, *custodian* [16] to run the electronic structure code (*e.g.*, VASP [1]) and correct job errors, and *pymatgen-db* to parse the output files and store the results in the **tasks** collection (as described previously). Thus, upon submission of a compound, automated processes will carry out the calculations, resulting in calculated data appearing in the **tasks** collection.

The calculation workflow only populates the **tasks** collection. The **materials** collection and higher level **analysis** data are generated by a series of *builders*. The builders generally operate in a MapReduce style; for example, the materials builder collects all tasks pertaining to a single material and builds one document for that material that combines properties computed from all tasks. The builders also resolve conflicts; for example, if two tasks contain the energy for a compound, the builder chooses the energy that is converged more accurately (*e.g.*, large *k*-point mesh or tighter energy convergence) or the magnetic state with the lowest energy.

3. The Materials API

The Materials Project RESTful API allows users to directly access Materials Project data via the Hypertext Transfer Protocol (HTTP), and provides an efficient way for users to programmatically query for materials information instead of relying on browser-based interfaces. The Materials API is designed using the REST Architectural Style [23], thus leveraging widely deployed HTTP infrastructure and related standards. Like the APIs of many well-known sites (*e.g.*, Netflix, Dropbox), the Materials API uses some shared knowledge about the form and semantics of Uniform Resource Identifiers (URIs), as well as pre-determined media types, to reduce the number of round-trips needed to discover and use the interface. By convention, this deviation from the full set of requirements of a REST API are indicated by the term "RESTful".

Under RESTful design, each object is represented as a unique resource and can be queried in a uniform manner. A RESTful HTTP service exposes a consistent set of semantics that uses HTTP methods (GET, POST, PUT, DELETE, etc.) in conjunction with unique URIs to access the underlying resources. This allows for the creation of an API using a combination of HTTP methods and URIs. For the purposes of the Materials API, this means that each document or object (such as a **task**, **material** or **analysis**) can be represented by a unique URI and an HTTP verb can be used to act on that object. In most cases, this action returns structured data that represents the object or the result of an operation against the object. In a RESTful design, the HTTP media-type indicates the type and format of the object; in the Materials API, we consistently use JSON with a media type "application/json".

3.1. URL design

Fig. 3 shows the general URL format for the Materials API. The URL format is designed to be simple and intuitive, and generally comprises five main components:

1. The first part of the URL (https://www.materialsproject.org/rest/v2) is the preamble. The "v2" at the end of the preamble denotes that this is currently version 2 of the API, and provides flexibility for future improvements to the API (including backwards incompatible versions) while continuing to support applications/analyses built on earlier versions.

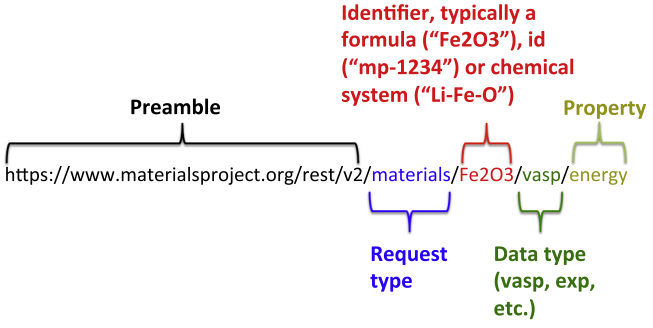


Fig. 3. Example of a URL format for the Materials API.

Table 1
Examples of supported resource keywords in the Materials API.

Resource keyword	Description	Supported identifiers	HTTP methods
materials	Information about a material or a set of materials, *e.g.*, energies, structure parameters, etc.	materials ids (*e.g.*, "mp-1234"), formulas (*e.g.*, "Fe2O3"), chemical systems (*e.g.*, "Li–Fe–O")	GET
tasks	Information about a task, *e.g.*, calculation parameters, etc.	task ids (*e.g.*, "mp-1234")	GET
phase_diagram	Phase diagram data for a chemical system	Chemical system (*e.g.*, "Li–Fe–O")	GET
reaction	Information about a reaction, *e.g.*, balanced reaction, reaction enthalpy	Arrays of reactants and products (*e.g.*, "reactants[] = MgO&reactants[] = Al_2O_3 &products[] = $MgAl_2O_4$")	GET
battery	Information about a battery material, *e.g.*, voltage, capacity, etc.	battery ids (*e.g.*, "mp-300019017"), formulas (*e.g.*, "LiFePO4")	GET
query	Provides for highly flexible queries based on the MongoDB syntax	No identifiers supported. Instead, two string parameters representing a query criteria and requested properties have to be supplied via POST. *E.g.*, "{"criteria": "{'nelements': 2}", "properties": "['formation_energy_per_atom']"}	POST
snl	Allows users to submit new structures for calculation by the Materials Project. Currently in beta testing with a limited group of users	No identifiers supported. Structures are submitted as a well-defined JSON string format that allows users to supply provenance (*e.g.*, publications to be cited, etc.)	POST/ GET

2. The next part of the URL ("materials") is a resource keyword that denotes the resource type requested. In the example shown in Fig. 3, this indicates that the request is for data about a material or set of materials. Other supported keywords and their functions are outlined in Table 1.
3. The resource keyword is followed by an identifier. As discussed in the previous section, the Materials Project assigns unique identifiers ("ids") to materials, computational tasks, etc. Besides these unique identifiers (which correspond to a single entity), some resource keywords also support other identifiers which allow users to easily perform common queries for entire sets of materials. The example shown in Fig. 3 is a request for data for all polymorphs with formula Fe_2O_3. For the "materials" keyword, another supported identifier is a "–" separated list of element symbols, which allows a user to request data on all materials in a chemical system.
4. The remainder of the url ("/vasp/energy") specifies the type of data requested. The supported options for this last part are highly dependent on the resource keyword. Some resources do not require any data specification at all. For the "materials" resource example, the request is for the energy of the materials computed using the Vienna Ab initio Simulation Package (VASP) [1].

In general, RESTful HTTP APIs prescribe the use of the GET HTTP method for idempotent, read-only queries and the POST HTTP method creation of new resources (non-idempotent). Currently, most of the idempotent Materials API calls use the GET http method, with the exception of "query" which requires the use of POST due to potentially large query strings that exceed browser/server data size limits for the GET method. This is a fairly common pattern in RESTful HTTP APIs when dealing with large inputs, to work around the limitations of the protocol. The "snl" call, a beta feature which allows users to submit new structures to the Materials Project for calculation (and hence is non-idempotent), uses the POST method for structure submission, but the GET method for looking up information about a submission. It should be noted that Table 1 only lists frequently used resource keywords and examples and is not a comprehensive listing of all supported resources.

3.2. Response formats

The Materials API uses the JavaScript Object Notation (JSON) as the primary format for responses. The JSON was selected as it is an extremely lightweight format with parser support in almost all common programming languages. The API also supports the common XML and YAML formats, which the user can specify via a *format* GET/POST parameter.

An example of the truncated JSON response for the request in Fig. 3 is given in Fig. 4. Besides the actual data requested (under the "response" key), the complete response also includes metadata such as the date the response was generated and the versions of the pymatgen code, database, and REST interface used to generate the response. These metadata are important for data provenance, given that the database as well as the supporting code infrastructure are constantly evolving.

3.3. Security and API keys

All requests to the Materials API must be done over Secure Hypertext Transfer Protocol (HTTPS) for security reasons. Most requests require the use of an API key, which users can obtain


```
{
    created_at: "2014-07-18T11:23:25.415382",
    valid_response: true,
  - version: {
        pymatgen: "2.9.9",
        db: "2014.04.18",
        rest: "1.0"
    },
  - response: [
      - {
            energy: -67.16532048,
            material_id: "mp-24972"
        },
      - {
            energy: -132.33035197,
            material_id: "mp-542309"
        },
      + {…},
      + {…},
      + {…},
      + {…},
      + {…},
      + {…},
      + {…},
      + {…}
    ],
    copyright: "Materials Project, 2012"
}
```


Fig. 4. Typical response format for the Materials API. The response has been truncated for brevity.

through their Materials Project dashboard (https://www.materialsproject.org/dashboard). The API key can be specified either as an *API_KEY* GET/POST parameter, or as an *x-api-key* header. The use of a unique API key for each user provides an efficient mechanism for the implementation of API features that require user identification, *e.g.*, the submission of structure prediction requests or computation requests.

4. High level implementation in Python Materials Genomics

In recent years, Python has become one of the most popular programming languages for scientific computing. This is in no small part due to its highly readable syntax, large standard library, as well as the establishment of high-performance numerical and scientific libraries such as Numpy and Scipy [27,28]. Most of the Materials Project's open-source software stack is implemented in Python. In particular, the Python Materials Genomics (pymatgen) library [15] powers most of the materials analyses in the Materials Project, providing core object definitions, and a well-tested set of structure and thermodynamic analysis tools relevant to many applications.

To make it easier for users to use the Materials API, a high-level interface to the API known as the *MPRester*, has been implemented in *pymatgen*'s *matproj.rest* module. Using this interface, users can obtain data through the Materials API with a minimal amount of coding and further analyze that data using the many tools available in A̋s a simple example, only four lines of code are necessary to obtain all structures corresponding to a formula or chemical system from the Materials Project and write them to files in the Crystallographic Information File (CIF) format, as follows:

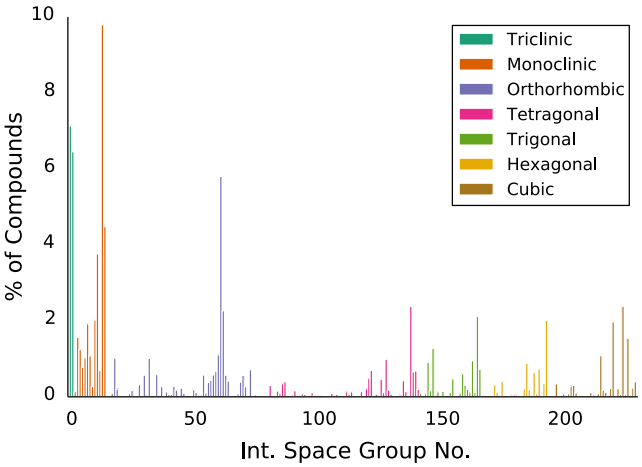
```
from pymatgen import MPRester, write_structure
with MPRester("USER_API_KEY") as mr:
    # Get all structures in the Fe-O system (~64) by
  making a call to
    # https://www.materialsproject.org/rest/v2/
  materials/Fe-O/structures
  structures = mr.get_structures("Fe-O")
    for i, struct in enumerate(structures):
        write_structure(struct, "%s-%d.cif" %
  (struct.formula, i))
```

In the following sections, we will demonstrate a few examples of more sophisticated analyses that can be performed using the Materials API. Most of these analyses were performed using the *pymatgen* high-level interface to the Materials API.

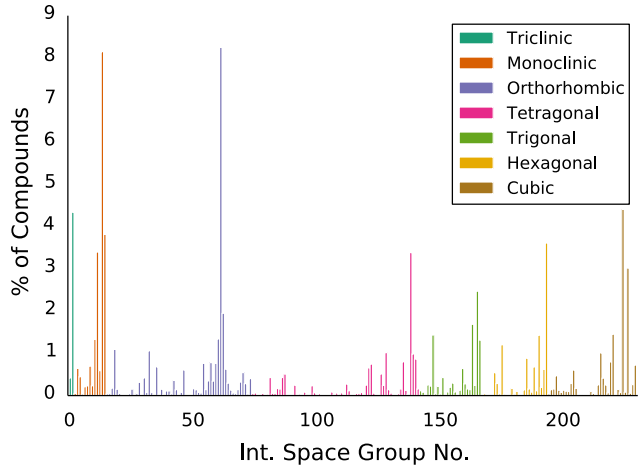
5. Usage examples

5.1. Datamining materials data

Using the Materials API, a user can programmatically query for data on a large number of materials, and perform data mining to determine trends. An example of this kind of analysis is given in Fig. 5. In Fig. 5a, a histogram of the distribution of space groups for all calculated materials in the Materials Project database that also belongs in the ICSD is constructed. Unlike many other databases which contain duplicate entries for the same structure, the Materials Project database performs matching of structures using a robust algorithm implemented in *pymatgen* [15]. This algorithm has been successfully used to group the 150,000 + structures in the ICSD into 50,000 + unique structures (including disordered structures). A similar analysis of space group distribution based on a compilation pf 34,692 structures by Baur and Kassner [29]



(a) Materials Project 2014 - 33,604 unique inorganic structures from the ICSD.



(b) Baur 1992 - 34,692 inorganic structures.

Fig. 5. Distribution of space groups for inorganic structures.

in 1992 is given in Fig. 5b. The Supplementary Information contains the actual Python code used to generate these figures in the form of an IPython notebook.

There is fairly good agreement in the distribution of the space groups between the Baur 1992 compilation and the Materials Project 2014 dataset. Both datasets find the two most common space groups to be $P2_1/c(14)$ and $Pnma(62)$, though the Materials Project 2014 dataset have a significantly higher percentage of structures in the $P2_1/c(14)$ space group (10.72%) compared to $Pnma(62)$ (7.6%), while the Baur dataset have similar percentages for both space groups (8.1–8.2%). The high symmetry space groups such as $Fm\bar{3}m(255)$ and $P6_3/mmc(194)$ form a slightly higher percentage of the Baur dataset compared to the Materials Project dataset. It should be noted that the Materials Project data only contains ordered structures, while the Baur compilation may contain both ordered and disordered structures, which may explain the slightly higher incidence of higher symmetry space groups in the Baur dataset.

5.2. Efficient materials computations and analyses

Using the Materials API, one can also obtain the relaxed structural parameters for most inorganic materials, which can serve as

the starting point for other kinds of property calculations. For example, many computations using higher-order methods such as the Heyd–Scuseria–Ernzerhof (HSE06) hybrid functional [30,31] or the GW method [32] generally start from the output of standard DFT computations (*e.g.*, utilizing the charge densities or wave functions computed). By querying for pre-relaxed structures in the Materials Project database, these calculations can be performed much more efficiently.

In addition, certain comparative materials analyses require the amalgamation of data on a large number of materials. For example, to determine the phase stability of a new material, one has to compare its free energy relative to that of competing phases. For complex materials comprising more than 3 elements, this requirement results in a combinatorial explosion in the number of calculations that need to be performed. Using the Materials API (and *pymatgen*'s phase diagram package), the researcher only needs to perform a single calculation of the phase of interest using parameters that are compatible with the Materials Project, and query for energetic data on the other materials in the relevant chemical systems from the Materials Project database. An example of this kind of analysis is presented in the authors' previous article on the *pymatgen* package [15].

6. Community development

A key objective of the Materials API is to build a community of developers that create new applications utilizing Materials Project data. In fact, the Materials API today powers most of the Materials Project's collaborative endeavors that comprise large numbers of scientists/ developers across many institutions. For example, there are several ongoing collaborations to expand the suite of properties available in the Materials Project, including elastic constants, phonons and electronic transport properties.

We are also in process of further extending the Materials API beyond simple data requests to support user contributions of data. This feature is already available to a limited set of users. For example, Castelli et al. [33] recently contributed band gaps for 2378 materials calculated using the GLLB-SC functional [34], which has been shown to provide more accurate band gaps for semiconductors and insulators at a computational cost that is commensurate with standard semi-local DFT [35]. An example can be seen at https://www.materialsproject.org/materials/mp-1143/. A well-defined data format has already been developed in the *pymatgen* library, with options for providing provenance on supplied data (*e.g.*, works to be cited, code used to generate data, etc.), and all contributions are given proper acknowledgment. By enabling community contributions, the Materials Project aims to become a robust repository for materials information that is not limited by the computational and human resources of any single group.

Yet another planned community development effort powered by the Materials API is the "Materials Genomics Cloud" (MGCloud), a cloud compute, storage and analysis platform for materials. The Materials API will be the primary communication link between the MGCloud and the Materials Project database. For example, the MGCloud will interact with the Structure Predictor app of the Materials Project to guide users in creating reasonable novel structures, and also check submitted structures against the entire Materials Project database. "Instant" answers can be provided if data already exists within the Materials Project database without the need for further computationally expensive calculations. Analyses that require consolidation of data for multiple materials (*e.g.*, phase stability) can be provided with minimal additional calculations by integrating data from the Materials Project. The MGCloud is already undergoing beta testing with a limited set of users today, and will soon be released.

7. Conclusion

The Materials Genome Initiative [36,37] has emphasized the importance of shared data collections in materials science and engineering. The Materials Project is one early example of the impact of MGI, providing a large amount of computed materials data coupled with powerful analysis tools and an open software stack. We expect such large materials databases to become increasingly commonplace. Providing programmatic interfaces to query such databases is key to enabling new analyses and applications. The Materials API for the Materials Project provides an example of such a programmatic interface built on REpresentational State Transfer (REST) principles. Coupled with the open source pymatgen materials analysis library, the Materials API has already found numerous applications in powering new collaborations and developing an active user community. It is hoped that this API implementation will also serve as a template for other materials databases, leading to more facile, open data access within the materials research community as a whole.

Acknowledgments

This work was supported by the Department of Energy's Basic Energy Sciences program under Grant No. EDCBEE. We also thank the National Energy Research Scientific Computing Center (NERSC), a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231, for providing invaluable computing resources and IT support for this project.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.commatsci.2014.10.037.

References

[1] G. Kresse, J. Furthmuller, Phys. Rev. B 54 (1996) 11169.
[2] M.J. Frisch, G.W. Trucks, H.B. Schlegel, G.E. Scuseria, M.A. Robb, J.R. Cheeseman, J.A. Montgomery Jr., T. Vreven, K.N. Kudin, J.C. Burant, J.M. Millam, S.S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G.A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J.E. Knox, H.P. Hratchian, J.B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R.E. Stratmann, O. Yazyev, A.J. Austin, R. Cammi, C. Pomelli, J.W. Ochterski, P.Y. Ayala, K. Morokuma, G.A. Voth, P. Salvador, J.J. Dannenberg, V.G. Zakrzewski, S. Dapprich, A.D. Daniels, M.C. Strain, O. Farkas, D.K. Malick, A.D. Rabuck, K. Raghavachari, J.B. Foresman, J.V. Ortiz, Q. Cui, A.G. Baboul, S. Clifford, J. Cioslowski, B.B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R.L. Martin, D.J. Fox, T. Keith, M.A. Al-Laham, C.Y. Peng, A. Nanayakkara, M. Challacombe, P.M.W. Gill, B. Johnson, W. Chen, M.W. Wong, C. Gonzalez, J.A. Pople, Gaussian 03, Revision C.02.
[3] X. Gonze, B. Amadon, P.-M. Anglade, J.-M. Beuken, F. Bottin, P. Boulanger, F. Bruneval, D. Caliste, R. Caracas, M. Côté, T. Deutsch, L. Genovese, P. Ghosez, M. Giantomassi, S. Goedecker, D.R. Hamann, P. Hermet, F. Jollet, G. Jomard, S. Leroux, M. Mancini, S. Mazevet, M.J.T. Oliveira, G. Onida, Y. Pouillon, T. Rangel, G.-M. Rignanese, D. Sangalli, R. Shaltaf, M. Torrent, M.J. Verstraete, G. Zerah, J.W. Zwanziger, Comput. Phys. Commun. 180 (2009) 2582.
[4] X. Gonze, G.M. Rignanese, M.J. Verstraete, J.M. Beuken, Y. Pouillon, R. Caracas, F. Jollet, M. Torrent, G. Zerah, M. Mikami, P. Ghosez, M. Veithen, J.Y. Raty, V. Olevano, F. Bruneval, L. Reining, R.W. Godby, G. Onida, D.R. Hamann, D.C. Allan, Zeitschrift für Krist. 220 (2005) 558.
[5] G. Hautier, A. Jain, S.P. Ong, B. Kang, C. Moore, R. Doe, G. Ceder, Chem. Mater. 23 (2011) 3495.
[6] G. Hautier, A. Jain, H. Chen, C. Moore, S.P. Ong, G. Ceder, J. Mater. Chem. 21 (2011) 17147.
[7] G. Ceder, G. Hautier, A. Jain, S.P. Ong, MRS Bull. 37 (2012) b1.
[8] S.P. Ong, V.L. Chevrier, G. Hautier, A. Jain, C. Moore, S. Kim, X. Ma, G. Ceder, Energy Environ. Sci. 4 (2011) 3680.
[9] Y. Mo, S.P. Ong, G. Ceder, Chem. Mater. 24 (2012) 15.
[10] J. Greeley, T.F. Jaramillo, J. Bonde, I.B. Chorkendorff, J.K. Nørskov, Nat. Mater. 5 (2006) 909.

[11] K. Yang, W. Setyawan, S. Wang, M. Buongiorno Nardelli, S. Curtarolo, Nat. Mater. 11 (2012) 614.
[12] J. Hachmann, R. Olivares-Amaya, S. Atahan-Evrenk, C. Amador-Bedolla, R.S. Sanchez-Carrera, A. Gold-Parker, L. Vogt, A.M. Brockway, A. Aspuru-Guzik, J. Phys. Chem. Lett. 2 (2011) 2241.
[13] A. Jain, S.P. Ong, G. Hautier, W. Chen, W.D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, K.A. Persson, APL Mater. 1 (2013) 011002.
[14] S. Curtarolo, W. Setyawan, G.L. Hart, M. Jahnatek, R.V. Chepulskii, R.H. Taylor, S. Wang, J. Xue, K. Yang, O. Levy, M.J. Mehl, H.T. Stokes, D.O. Demchenko, D. Morgan, Comput. Mater. Sci. 58 (2012) 218.
[15] S.P. Ong, W.D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V.L. Chevrier, K.A. Persson, G. Ceder, Comput. Mater. Sci. 68 (2013) 314.
[16] S.P. Ong, W. Richards, S. Dacek, X. Qu, A. Jain, Custodian (2014).
[17] A. Jain, Fireworks (2011).
[18] S.P. Ong, A. Jain, G. Hautier, B. Kang, G. Ceder, Electrochem. Commun. 12 (2010) 427.
[19] S.P. Ong, L. Wang, B. Kang, G. Ceder, Chem. Mater. 20 (2008) 1798.
[20] K.a. Persson, B. Waldwick, P. Lazic, G. Ceder, Phys. Rev. B 85 (2012) 1.
[21] G. Hautier, C.C. Fischer, A. Jain, T. Mueller, G. Ceder, Chem. Mater. 22 (2010) 3762.
[22] G. Hautier, C. Fischer, V. Ehrlacher, A. Jain, G. Ceder, Inorg. Chem. 656 (2010).
[23] R.T. Fielding, R.N. Taylor, ACM Trans. Internet Technol. 2 (2002) 115.
[24] P.W. Rose, B. Beran, C. Bi, W.F. Bluhm, D. Dimitropoulos, D.S. Goodsell, A. Prlic, M. Quesada, G.B. Quinn, J.D. Westbrook, J. Young, B. Yukich, C. Zardecki, H.M. Berman, P.E. Bourne, Nucleic Acids Res. 39 (2011) D392.
[25] R.H. Taylor, F. Rose, C. Toher, O. Levy, K. Yang, M. Buongiorno Nardelli, S. Curtarolo, Comput. Mater. Sci. 93 (2014) 178.
[26] G. Bergerhoff, R. Hundt, R. Sievers, I.D. Brown, J. Chem. Inf. Comput. Sci. 23 (1983) 66.
[27] E. Jones, T. Oliphant, P. Peterson, Others, {SciPy}: Open source scientific tools for {Python}.
[28] T.E. Oliphant, Comput. Sci. Eng. 9 (2007) 10.
[29] W.H. Baur, D. Kassner, Acta Crystallogr. Sect. B Struct. Sci. 48 (1992) 356.
[30] J. Heyd, G.E. Scuseria, M. Ernzerhof, J. Chem. Phys. 118 (2003) 8207.
[31] J. Heyd, G.E. Scuseria, M. Ernzerhof, J. Chem. Phys. 124 (2006) 219906.
[32] F. Aryasetiawan, O. Gunnarsson, Reports Prog. Phys. 61 (1998) 237.
[33] I.E. Castelli, F. Huser, M. Pandey, H. Li, K.S. Thygesen, A. Jain, K.A. Persson, G. Ceder, K.W. Jacobsen, Submiss. (2014).
[34] M. Kuisma, J. Ojanen, J. Enkovaara, T. Rantala, Phys. Rev. B 82 (2010) 1.
[35] I.E. Castelli, T. Olsen, S. Datta, D.D. Landis, S.r. Dahl, K.S. Thygesen, K.W. Jacobsen, Energy Environ. Sci. 5 (2012) 5814.
[36] T. Kalil, C. Wadia, Materials Genome Initiative for Global Competitiveness, Tech. Rep. June (NATIONAL SCIENCE AND TECHNOLOGY COUNCIL, 2011).
[37] G. Ceder, K. Persson, Sci. Am. 309 (2013) 36.