FireWorks for Materials Science

HPC Workflow

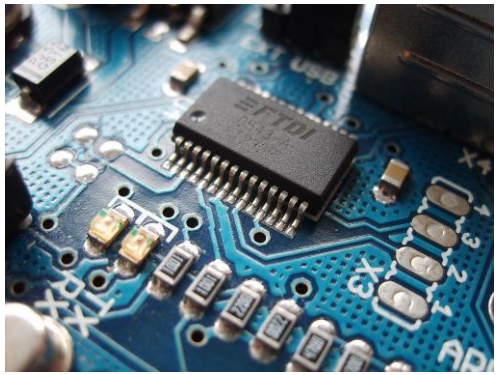
Guillaume Brunin

February 16, 2022

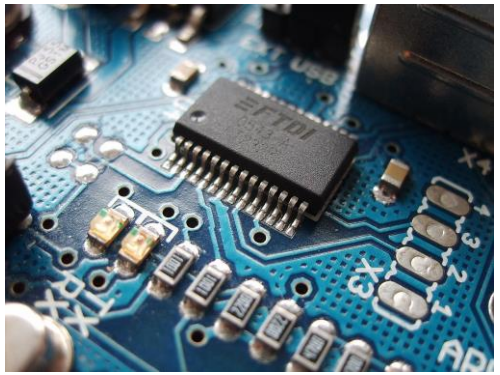
UCLouvain

Institute of Condensed Matter and Nanosciences

Functional materials are used in many applications every day

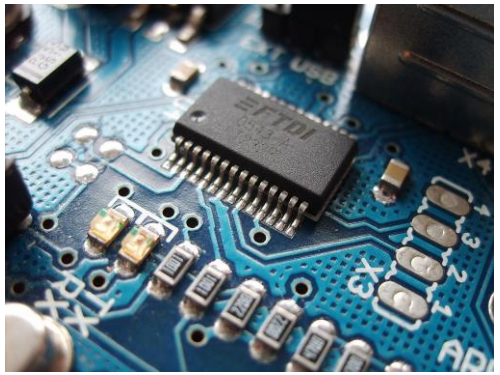


Functional materials are used in many applications every day



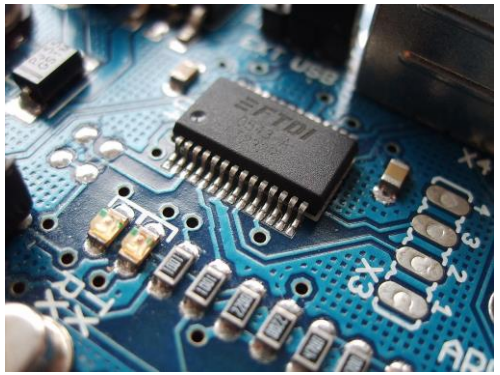
- Better materials for PV ?

Functional materials are used in many applications every day



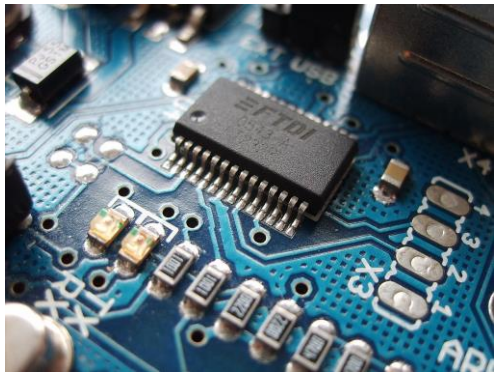
- Better materials for PV ?
- Faster, smaller transistors ?

Functional materials are used in many applications every day



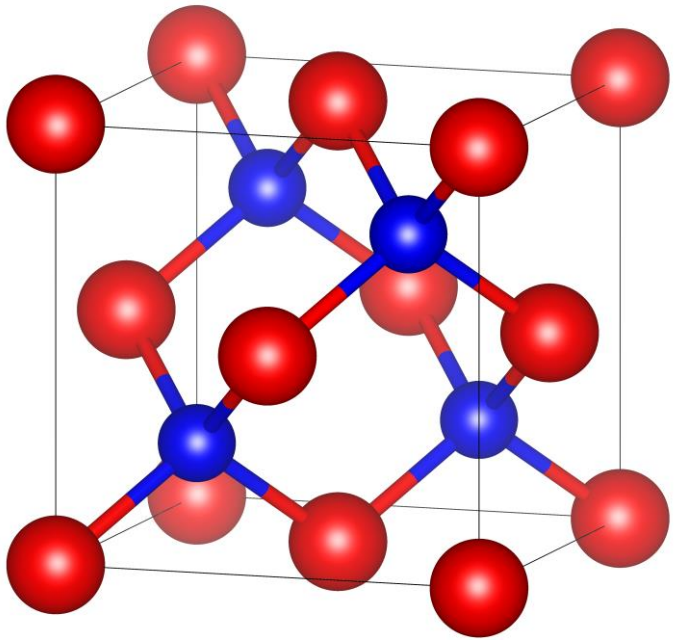
- Better materials for PV ?
- Faster, smaller transistors ?
- Transparent electronics for smart windows ?

Functional materials are used in many applications every day

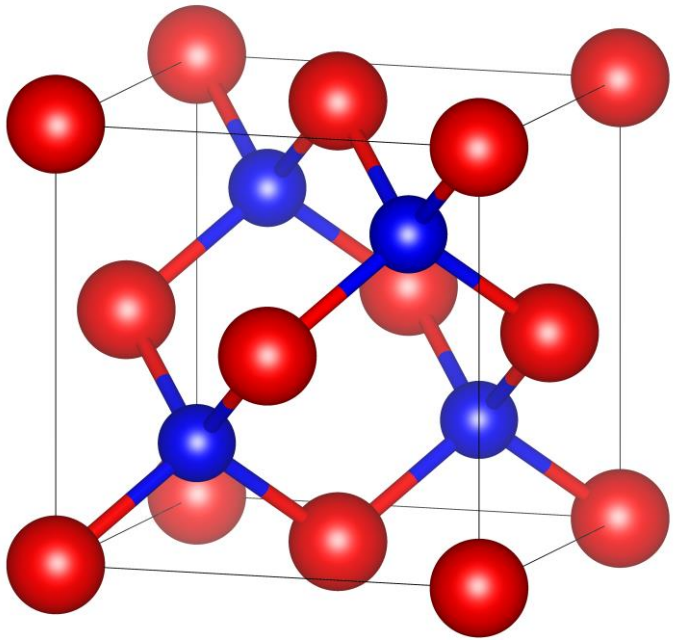


- Better materials for PV ?
- Faster, smaller transistors ?
- Transparent electronics for smart windows ?
- Better energy storage ?

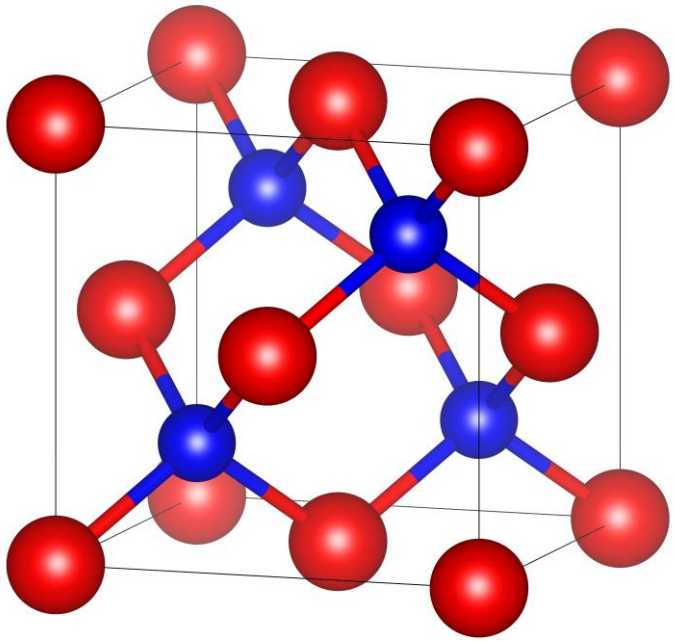
Computational predictions can help finding good new materials...



Computational predictions can help finding good new materials...

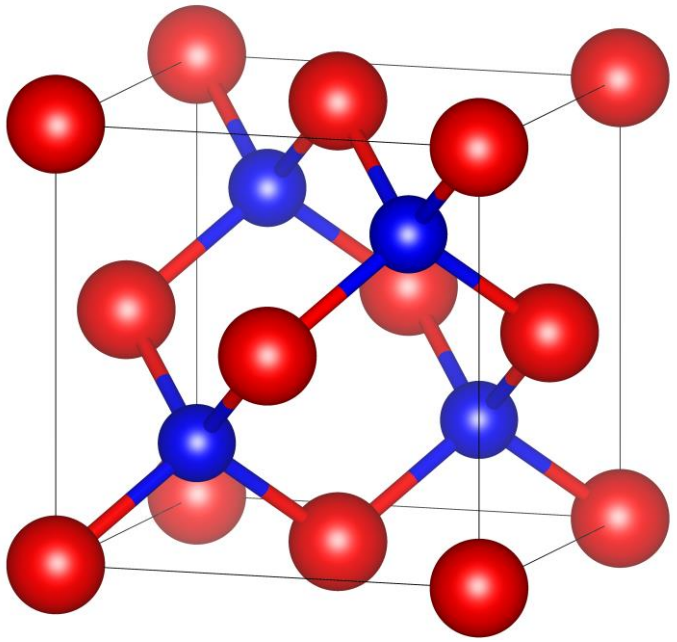


Computational predictions can help finding good new materials...



- Stability
- Thermodynamics, phase diagrams
- Electronic structure
- Transport (heat, electricity)
- Vibrational properties
- Magnetic properties
- Defect concentration
- ...

Computational predictions can help finding good new materials...

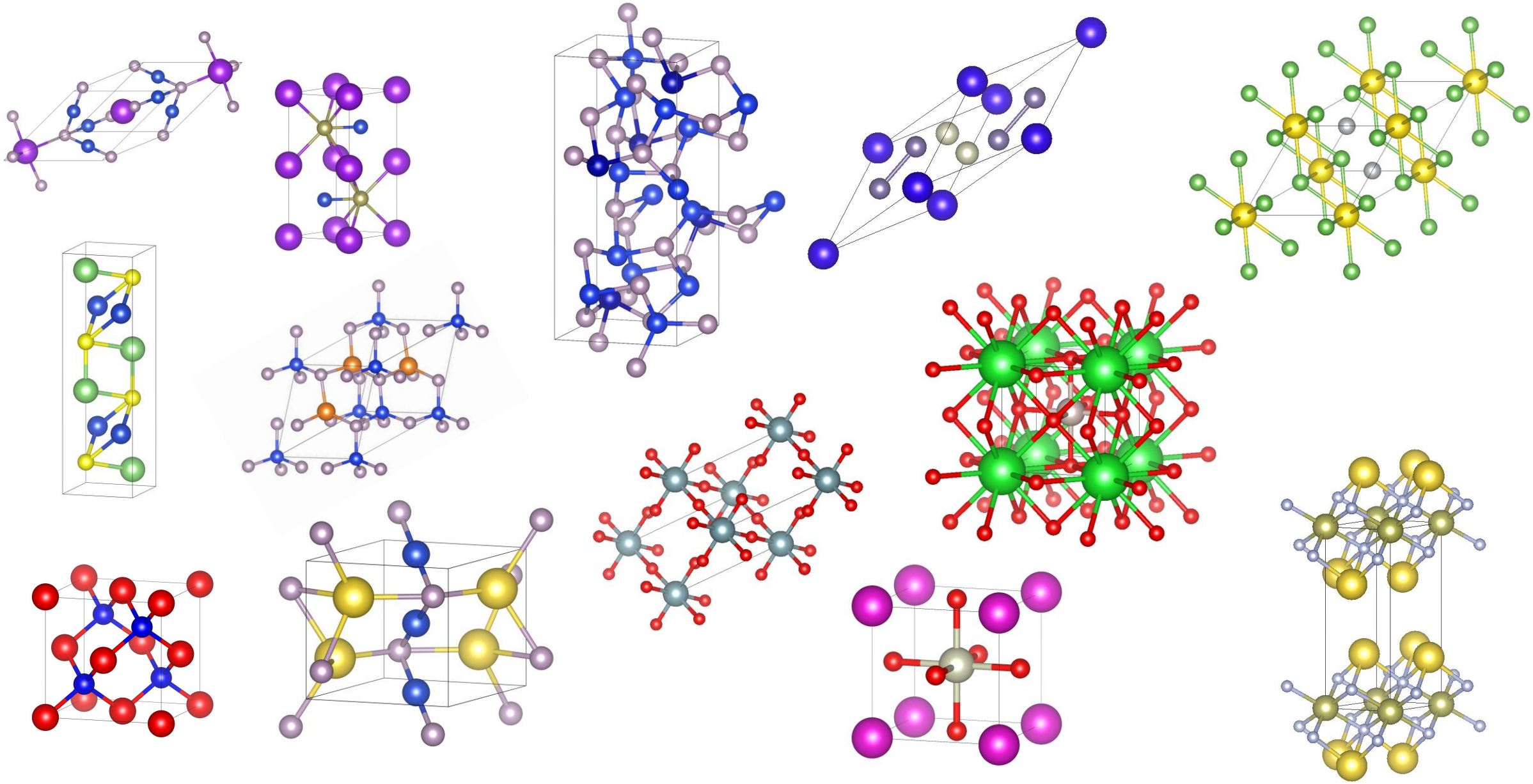


- Stability
- Thermodynamics, phase diagrams
- Electronic structure
- Transport (heat, electricity)
- Vibrational properties
- Magnetic properties
- Defect concentration
- ...

Each of these properties requires from 1 to 10's of different computations

... but we may have to search through 10,000’s of possibilities !

... but we may have to search through 10,000's of possibilities !



Instead of hiring an army of students, we can automate things

Instead of hiring an army of students, we can automate things

VS

```
for material in material_list:
    material.get_property()
    material.save()
```

Several steps need to be included in the process

```
for material in material_list:
    material.get_property()
    material.save()
```

This is beyond over-simplified !

Several steps need to be included in the process

- Input generation

abipy

pymatgen

```
for material in material_list:
    material.get_property()
    material.save()
```

This is beyond over-simplified !

Several steps need to be included in the process

- Input generation → abipy / pymatgen
- Workflow execution → FireWorks

```
for material in material_list:
    material.get_property()
    material.save()
```

This is beyond over-simplified !

Several steps need to be included in the process

- Input generation → abipy / pymatgen
- Workflow execution → FireWorks
- Error correction → Custodian / abipy

```
for material in material_list:
    material.get_property()
    material.save()
```

This is beyond over-simplified !

Several steps need to be included in the process

- Input generation → abipy / pymatgen
- Workflow execution → FireWorks
- Error correction → Custodian / abipy
- Data storage → mongoDB

```
for material in material_list:
    material.get_property()
    material.save()
```

This is beyond over-simplified !

Several steps need to be included in the process

- Input generation → abipy / pymatgen
- Workflow execution → FireWorks
- Error correction → Custodian / abipy
- Data storage → mongoDB
- Data analysis → abipy / pymatgen

```
for material in material_list:
    material.get_property()
    material.save()
```

This is beyond over-simplified !

Several steps need to be included in the process

- Input generation → abipy / pymatgen
- Workflow execution → FireWorks
- Error correction → Custodian / abipy
- Data storage → mongoDB
- Data analysis → abipy / pymatgen

```
for material in material_list:
    material.get_property()
    material.save()
```

This is beyond over-simplified !

Let’s assume we are all experts in the field and focus on FireWorks

- General-purpose workflow manager

Let’s assume we are all experts in the field and focus on FireWorks

- General-purpose workflow manager : could be used for many applications

Let’s assume we are all experts in the field and focus on FireWorks

- General-purpose workflow manager : could be used for many applications
- Automate calculations over arbitrary computing resources

Let’s assume we are all experts in the field and focus on FireWorks

- General-purpose workflow manager : could be used for many applications
- Automate calculations over arbitrary computing resources
- Support for several queueing systems (PBS, Slurm,...)

Let’s assume we are all experts in the field and focus on FireWorks

- General-purpose workflow manager : could be used for many applications
- Automate calculations over arbitrary computing resources
- Support for several queueing systems (PBS, Slurm,...)
- Clean and flexible Python API

Let’s assume we are all experts in the field and focus on FireWorks

- General-purpose workflow manager : could be used for many applications
- Automate calculations over arbitrary computing resources
- Support for several queueing systems (PBS, Slurm,...)
- Clean and flexible Python API
- Centralized database of jobs (MongoDB - could use something else later)

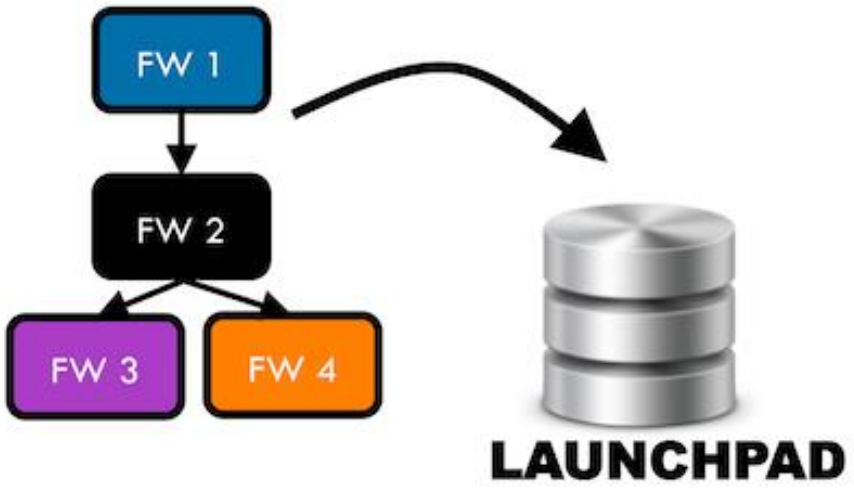
Let’s assume we are all experts in the field and focus on FireWorks

- General-purpose workflow manager : could be used for many applications
- Automate calculations over arbitrary computing resources
- Support for several queueing systems (PBS, Slurm,...)
- Clean and flexible Python API
- Centralized database of jobs (MongoDB - could use something else later)
- Support for dynamic workflows

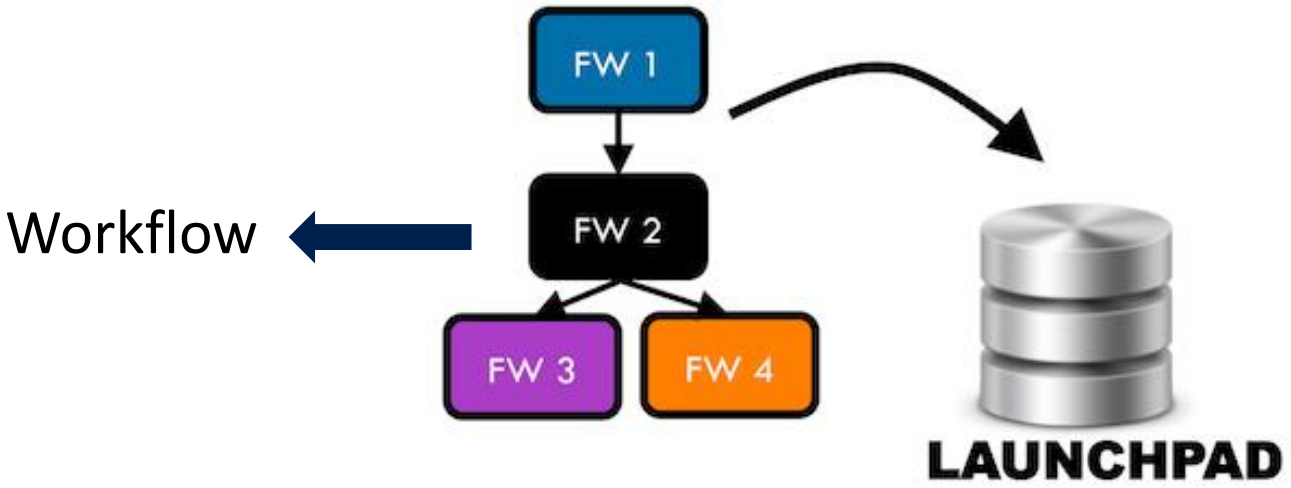
Let’s assume we are all experts in the field and focus on FireWorks

- General-purpose workflow manager : could be used for many applications
- Automate calculations over arbitrary computing resources
- Support for several queueing systems (PBS, Slurm,...)
- Clean and flexible Python API
- Centralized database of jobs (MongoDB - could use something else later)
- Support for dynamic workflows
- Web GUI monitor

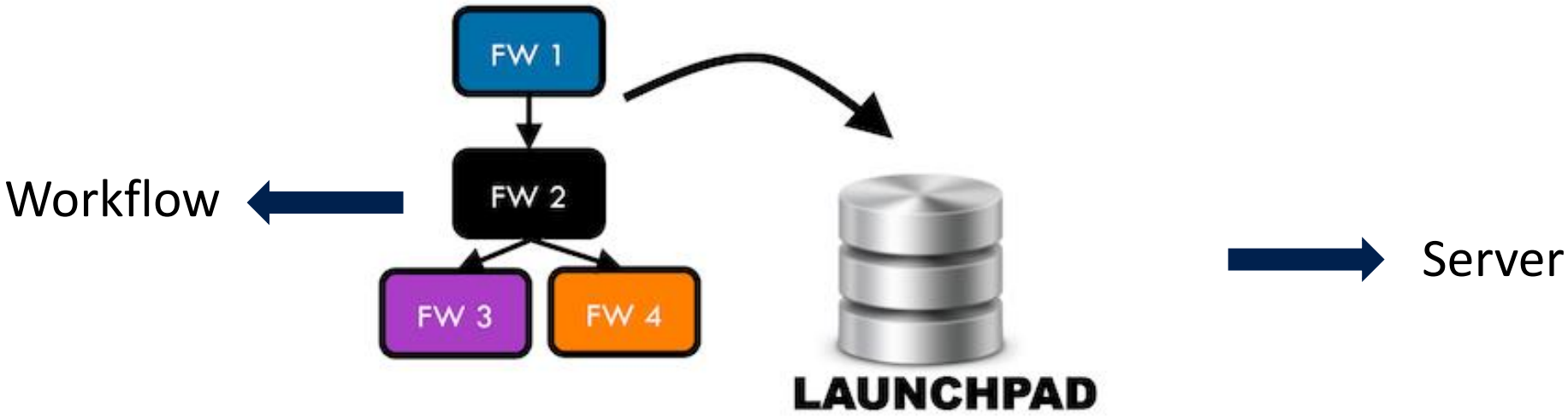
FireWorks is based on a centralized server



FireWorks is based on a centralized server

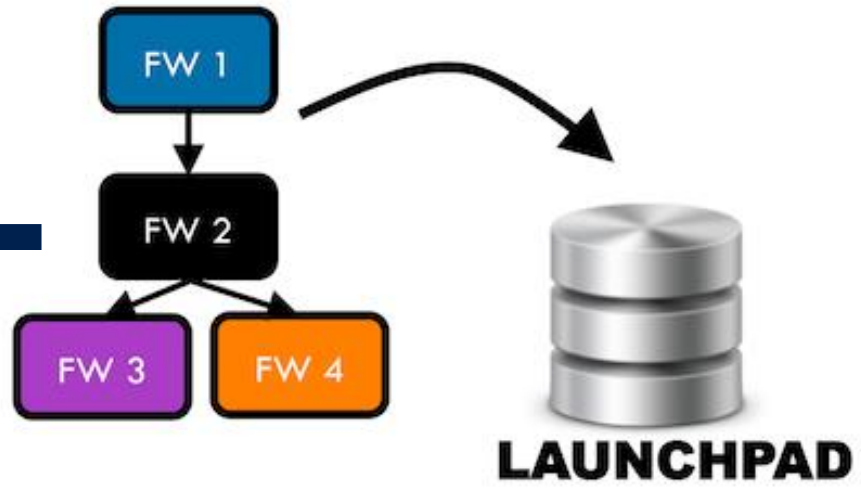


FireWorks is based on a centralized server



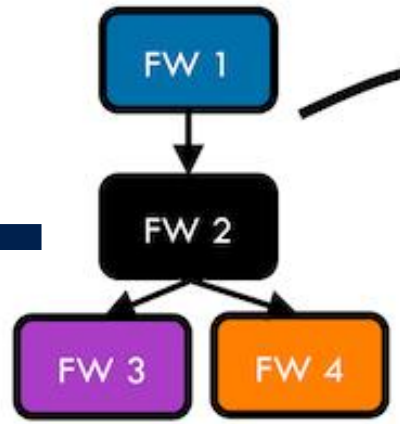
FireWorks is based on a centralized server

Workflow



Server : interface with the FW DB

FireWorks is based on a centralized server and on workers



Workflow

LAUNCHPAD

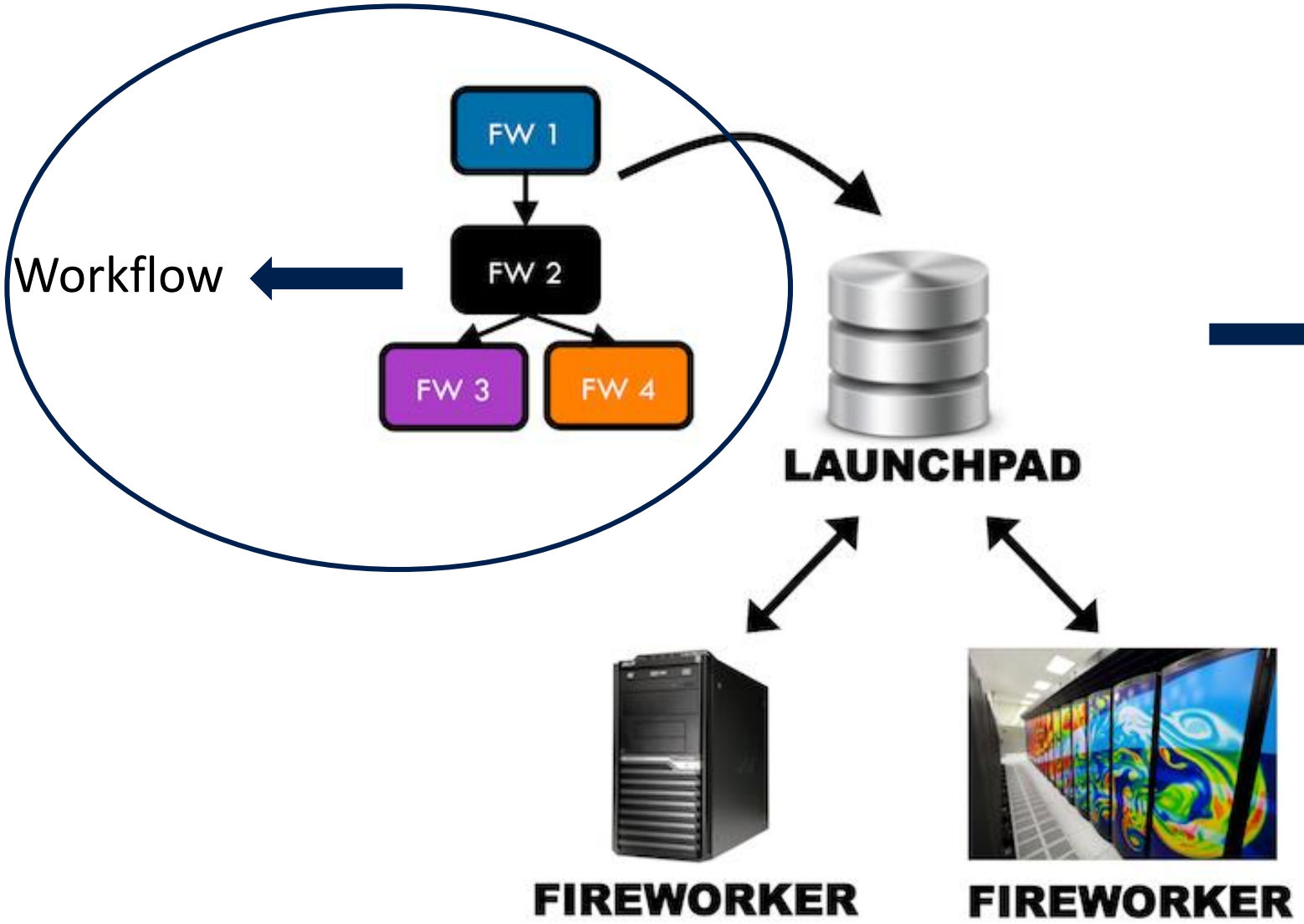
Server : interface with the FW DB

FIREWORKER

FIREWORKER

Workers : 1) execute the WF
2) stores the results in a DB

Let’s look a bit closer at how WF are defined in FireWorks

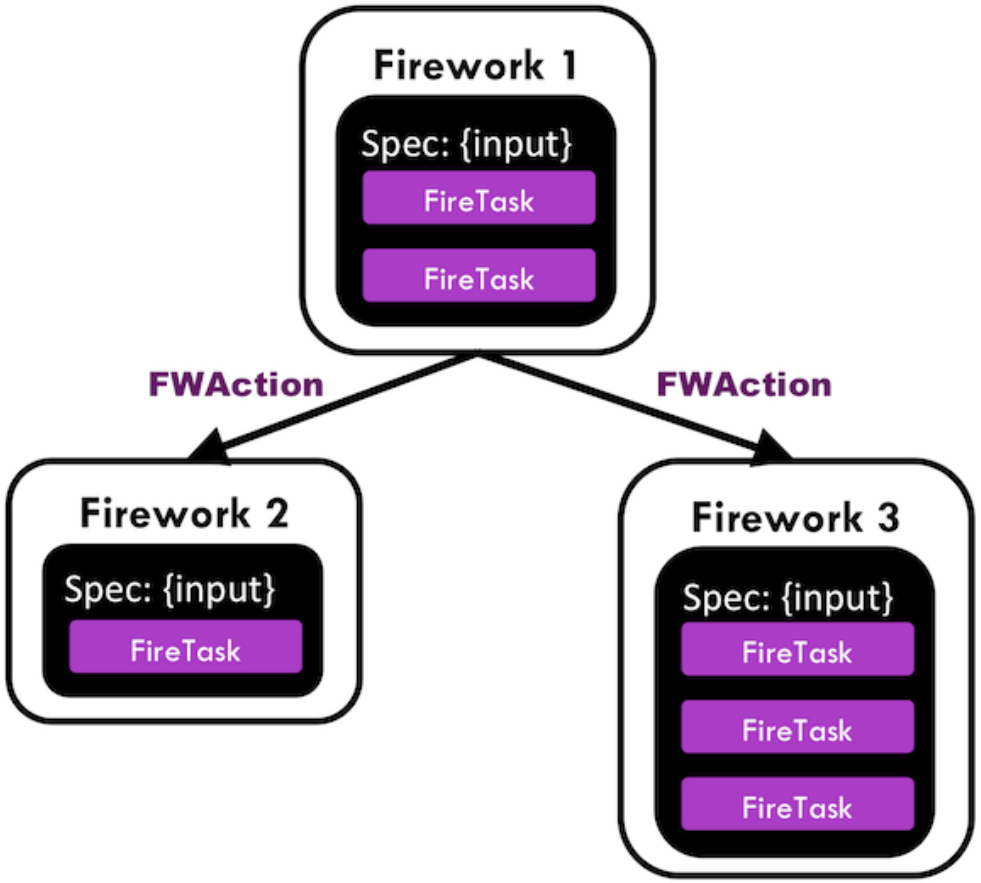


Server : interface with the FW DB

Workers : 1) execute the WF
2) stores the results in a DB

A FireWorks WF is composed of basic building units

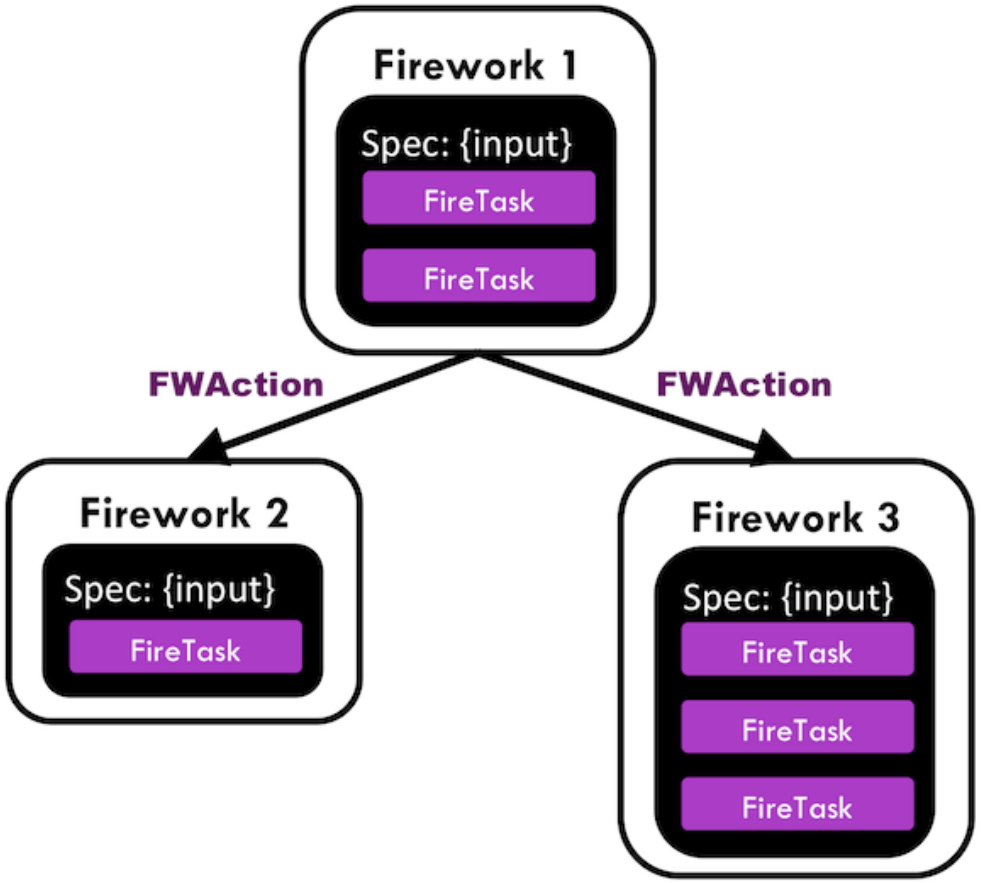
- Firework : list of tasks to be performed
  (~ 1 job in the queue)



A FireWorks WF is composed of basic building units

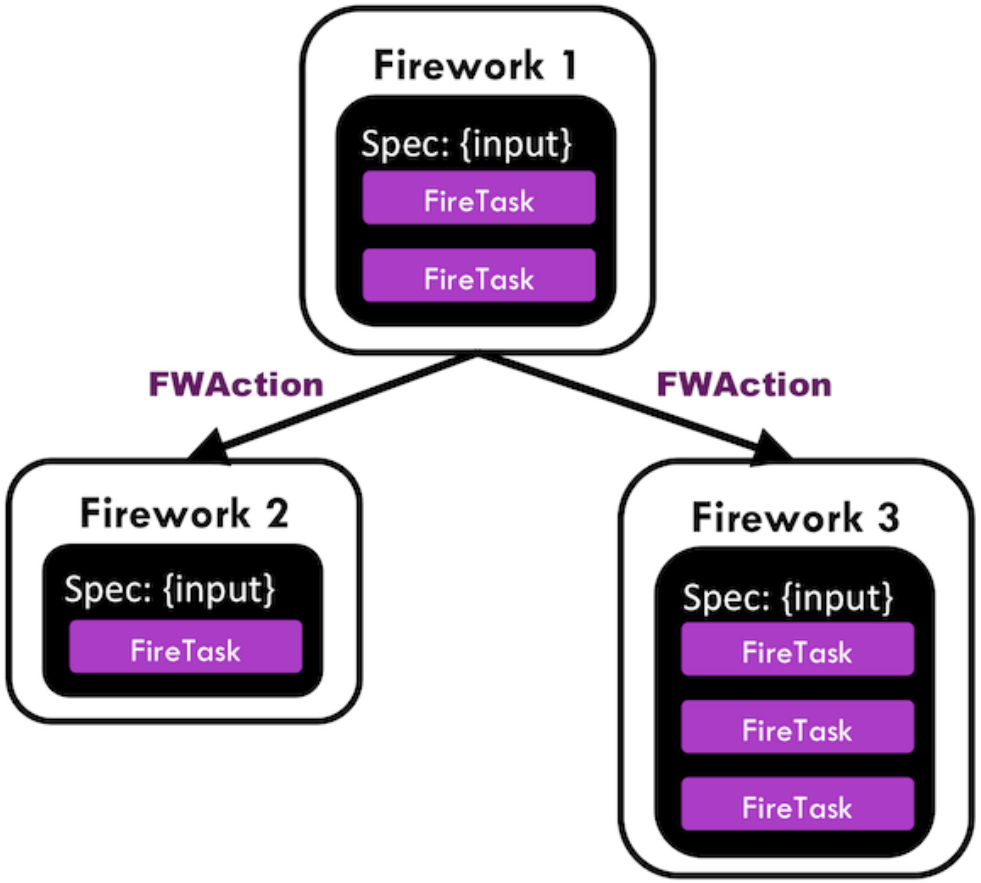
- Firework : list of tasks to be performed
  (~ 1 job in the queue)

- FireTask : atomic computing job
  (shell/python script, computation,
  copy, save result to DB,...)



A FireWorks WF is composed of basic building units

- Firework : list of tasks to be performed
  (~ 1 job in the queue)
- FireTask : atomic computing job
  (shell/python script, computation,
  copy, save result to DB,...)
- FWAction : object returned at the end of each
  FireTask

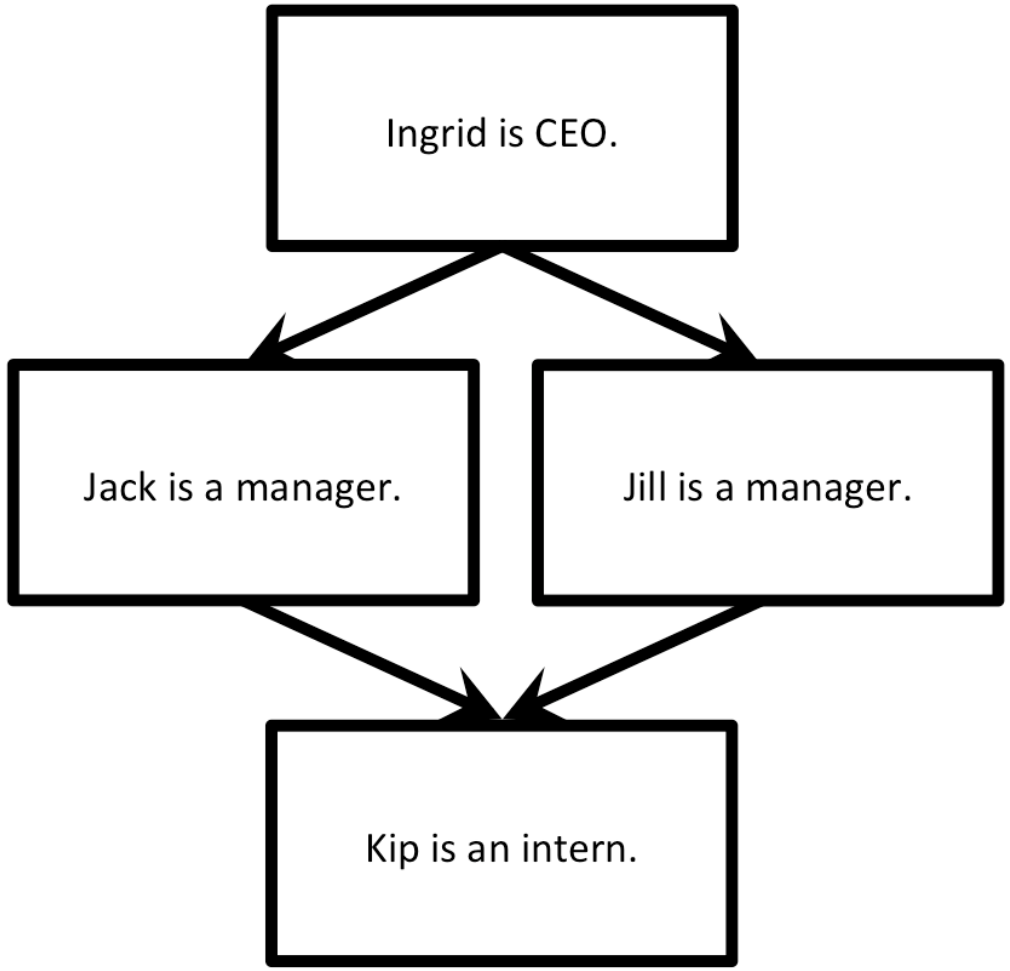


Let’s see a super simple example of a Workflow

```
# define four individual FireWorks used in the Workflow
task1 = ScriptTask.from_str('echo "Ingrid is the CEO."')
task2 = ScriptTask.from_str('echo "Jill is a manager."')
task3 = ScriptTask.from_str('echo "Jack is a manager."')
task4 = ScriptTask.from_str('echo "Kip is an intern."')

fw1 = Firework(task1)
fw2 = Firework(task2)
fw3 = Firework(task3)
fw4 = Firework(task4)

# assemble Workflow from FireWorks and their connections
workflow = Workflow([fw1, fw2, fw3, fw4],
                    {fw1: [fw2, fw3], fw2: [fw4], fw3: [fw4]})
```



Once the WF is added to the DB, we can tell FireWorks to start running

```
$ qlaunch rapidfire --nlaunches 1
INFO getting queue adapter
INFO Found previous block, usingblock_2019-02-04-16-25-31-583641
INFO The number of jobs currently in the queue is: 0
INFO 0 jobs in queue. Maximum allowed by user: 0
INFO Launching a rocket!
INFO Created new dir block_2019-02-04-16-25-31-583641/launcher_2019-02-04-16-27-54-081986
INFO moving to launch_dir block_2019-02-04-16-25-31-583641/launcher_2019-02-04-16-27-54-081986
INFO submitting queue script
INFO Job submission was successful and job_id is 5
INFO Launched allowed number of jobs: 1
```

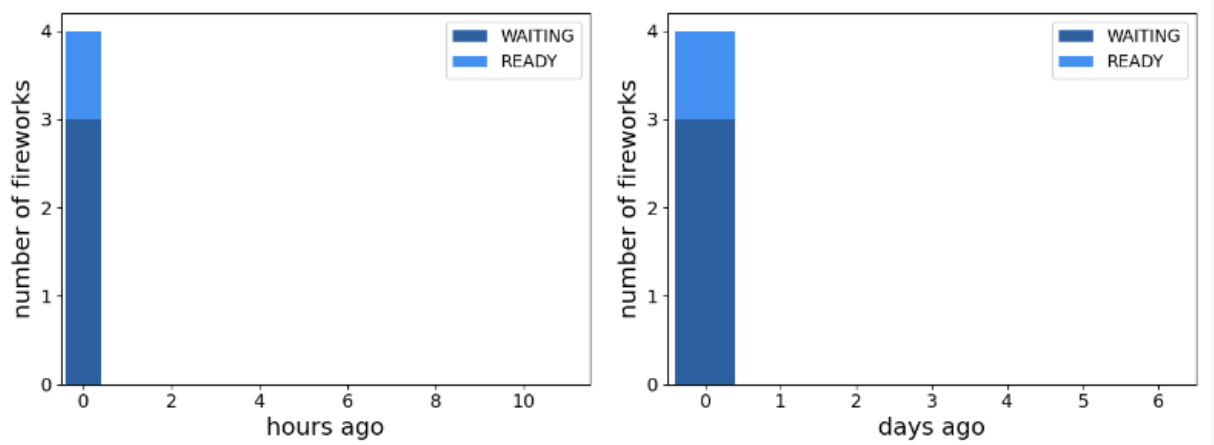
We can monitor what is happening using a GUI

FireWorks

Newest Workflows

CompanyStructure READY ID: 3088

AssignJob
AssignJob
AssignJob
AssignJob



Database snapshot

	Fireworks	Workflows

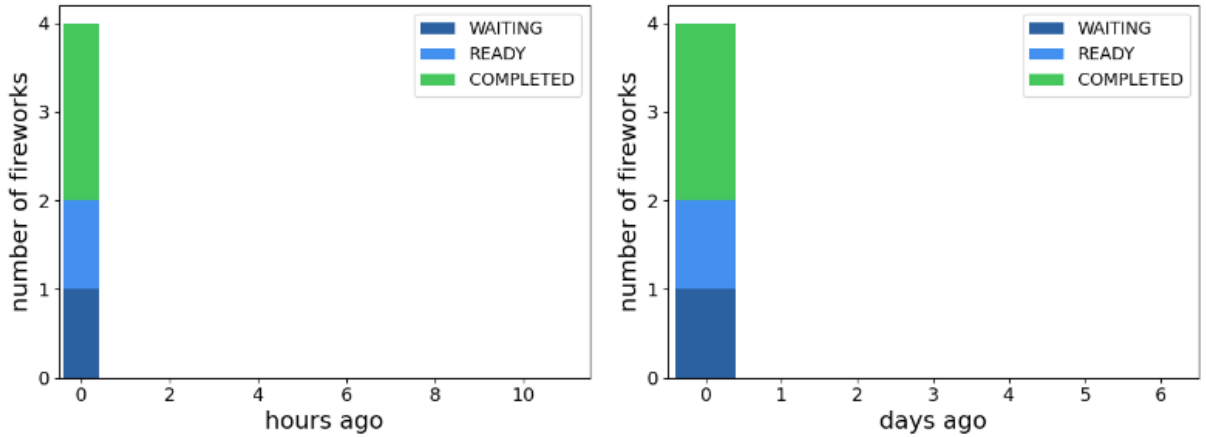
We can monitor what is happening using a GUI

FireWorks

Newest Workflows

CompanyStructure RUNNING ID: 3088

AssignJob
AssignJob
AssignJob
AssignJob



Database snapshot

	Fireworks	Workflows

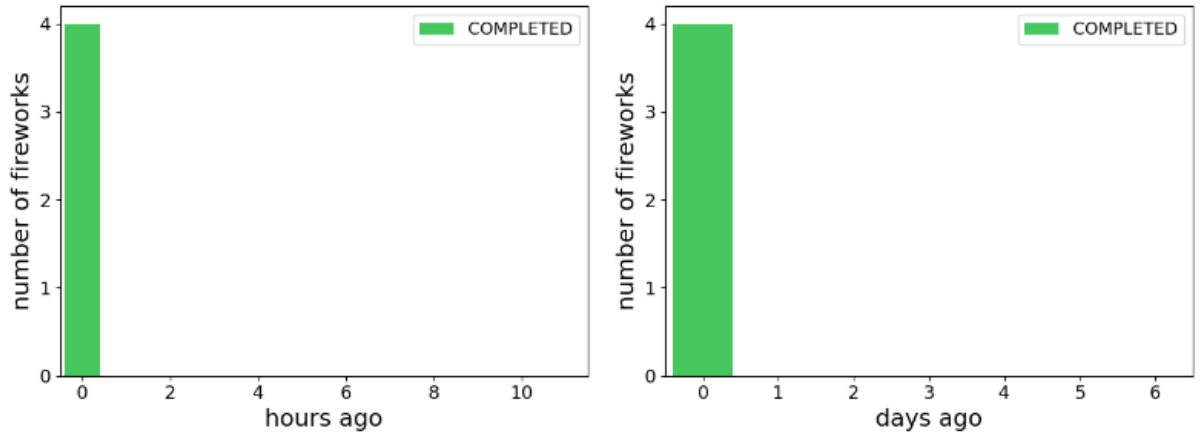
We can monitor what is happening using a GUI

FireWorks

Newest Workflows

CompanyStructure COMPLETED ID: 3088

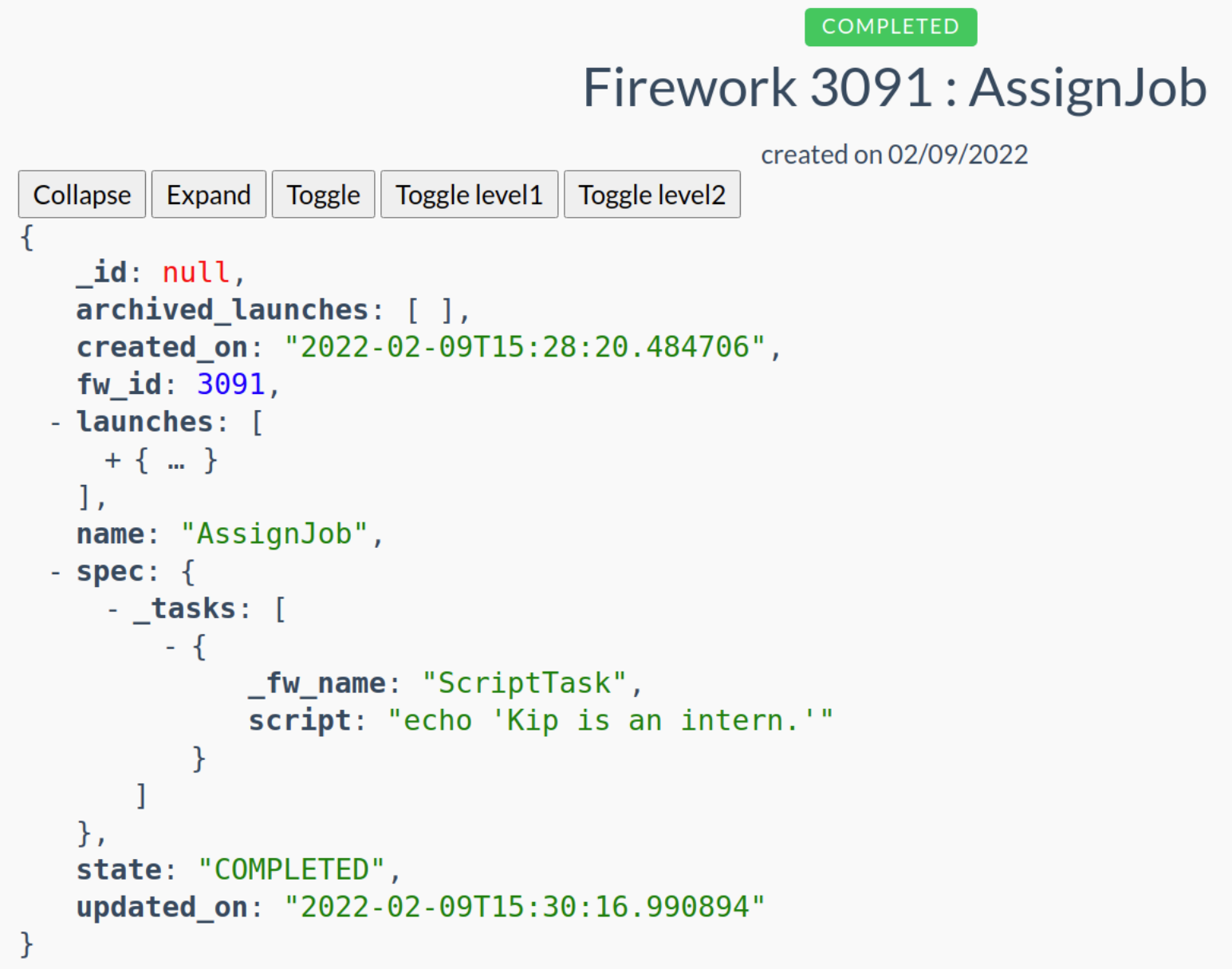
AssignJob
AssignJob
AssignJob
AssignJob



Database snapshot

	Fireworks	Workflows

We can monitor what is happening using a GUI



This is just a very superficial introduction !

The tutorials are very useful and the website well documented

- FireWorks documentation :

https://materialsproject.github.io/fireworks/

- Paper :

“FireWorks: a dynamic workflow system designed for high-throughput applications”. Concurr. Comput. Pract. Exp. 22, 5037–5059 (2015)

Several steps need to be included in the process

- Input generation → abipy / pymatgen
- Workflow execution → FireWorks
- Error correction → Custodian / abipy
- Data storage → mongoDB
- Data analysis → abipy / pymatgen

```
for material in material_list:
    material.get_property()
    material.save()
```

This is beyond over-simplified !

Most of these have been gathered into higher-level tools...

- Input generation → abipy / pymatgen
- Workflow execution → FireWorks
- Error correction → Custodian / abipy
- Data storage → mongoDB.
- Data analysis → abipy / pymatgen

```
for material in material_list:
    material.get_property()
    material.save()
```

This is beyond over-simplified !

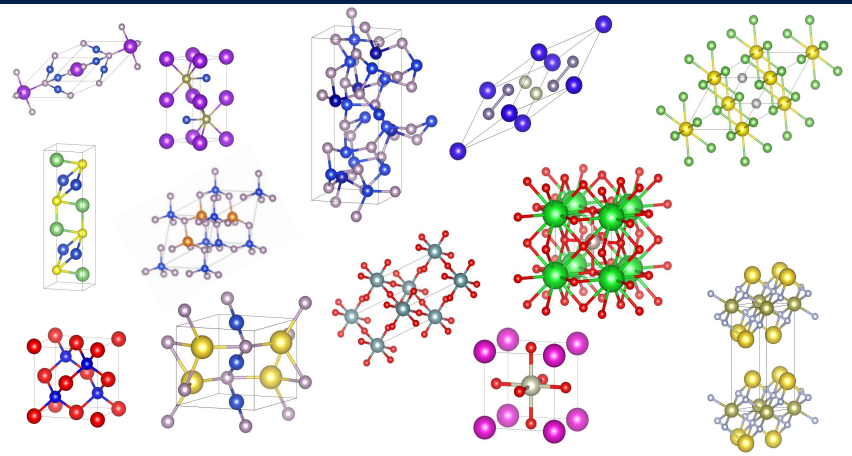
Most of these have been gathered into higher-level tools...

- Input generation → abipy, pymatgen
- Workflow execution → FireWorks
- Error correction → Custodian, abipy
- Data storage → mongoDB

→ abiflows, atomate

...designed for our material-science applications !

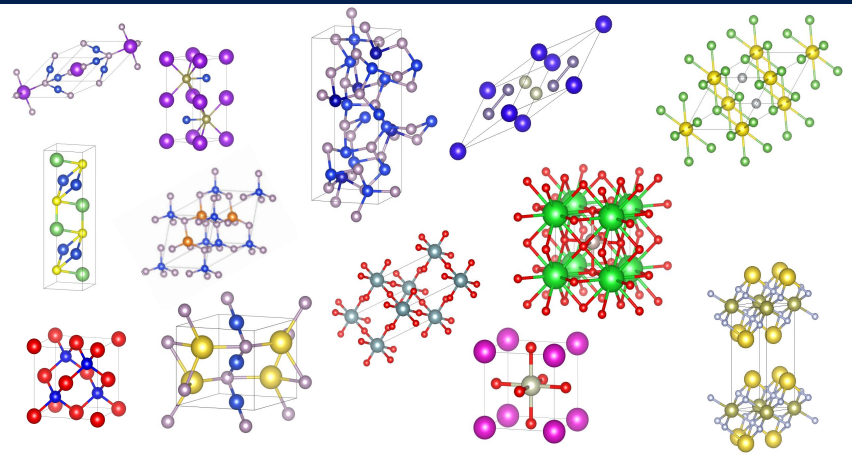
What can we do with that ? A few examples of what has been done



Characterization of the electronic transport in materials

"An ab initio electronic transport database for inorganic materials", F. Ricci et al., Scientific data 4, 170085 (2017)

What can we do with that ? A few examples of what has been done

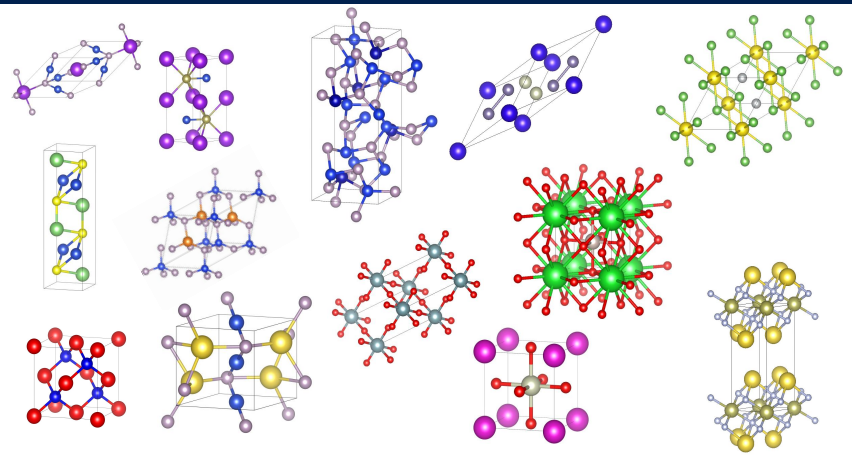


Characterization of the electronic transport in materials

FireWorks

"An ab initio electronic transport database for inorganic materials", F. Ricci et al., Scientific data 4, 170085 (2017)

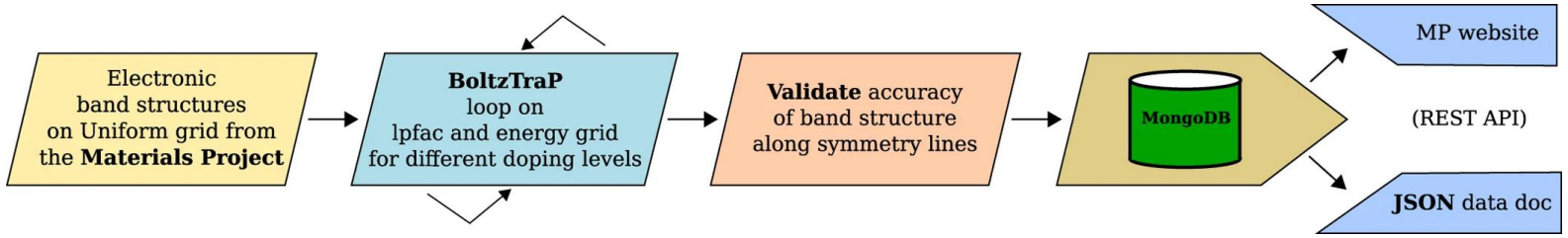
What can we do with that ? A few examples of what has been done



48000 materials

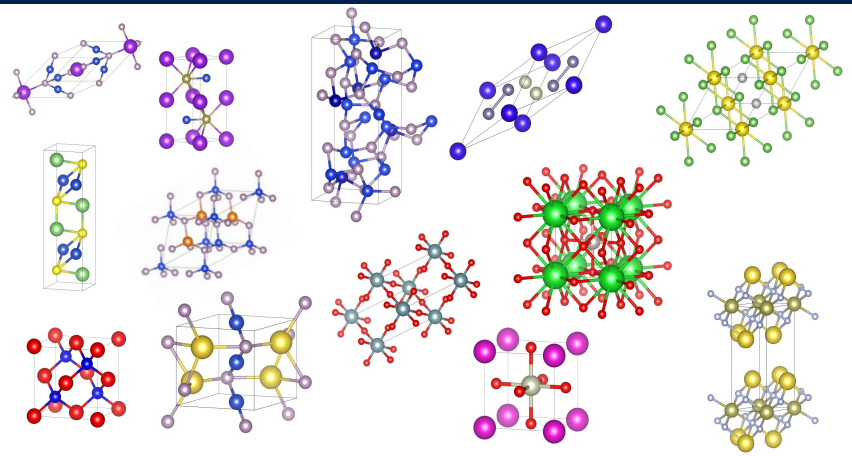
Characterization of the electronic transport in materials

FireWorks



"An ab initio electronic transport database for inorganic materials", F. Ricci et al., Scientific data 4, 170085 (2017)

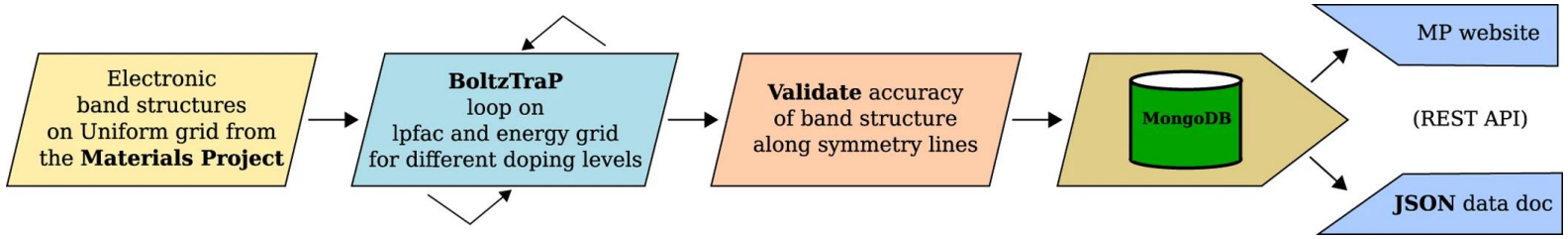
What can we do with that ? A few examples of what has been done



Characterization of the electronic transport in materials

FireWorks

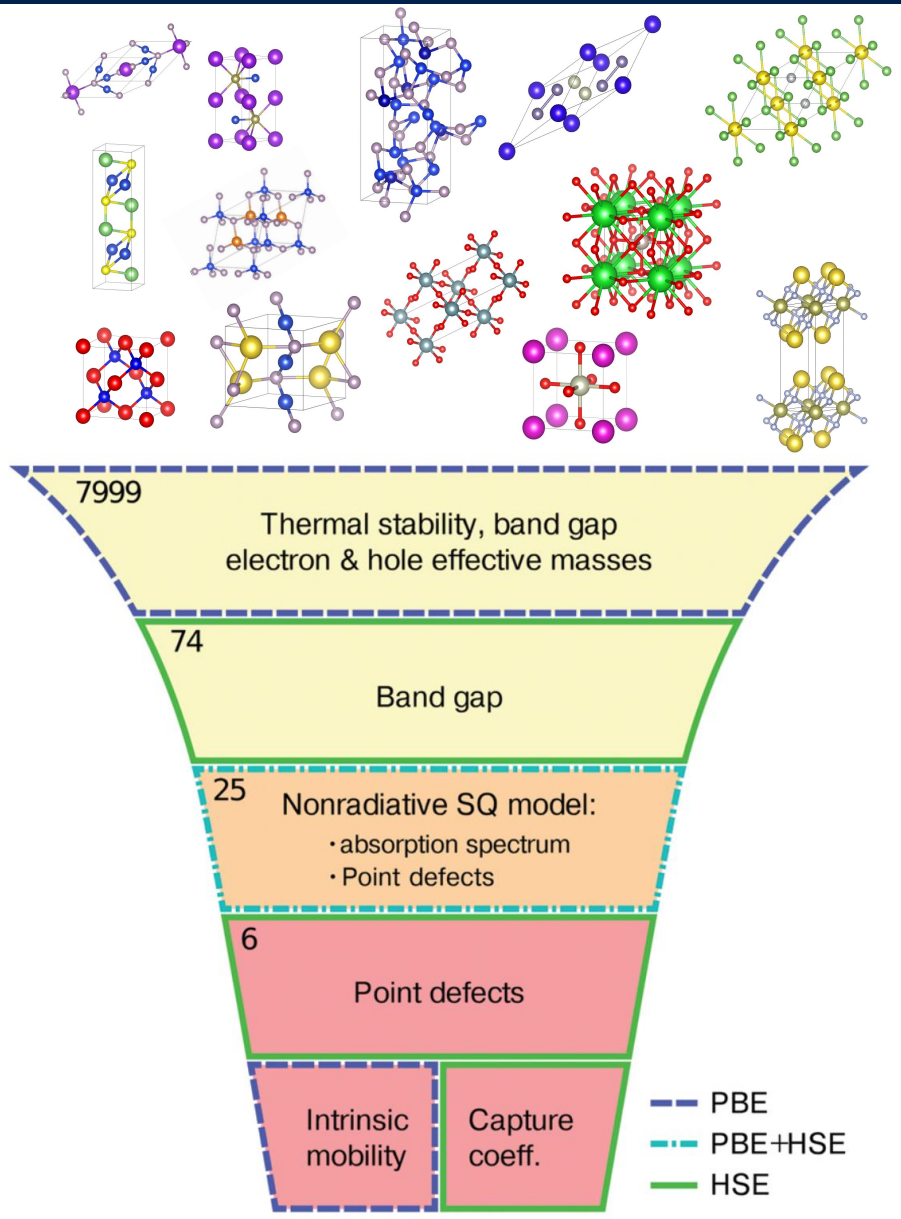
48000 materials



The results are now available in a database for all researchers !

"An ab initio electronic transport database for inorganic materials", F. Ricci et al., Scientific data 4, 170085 (2017)

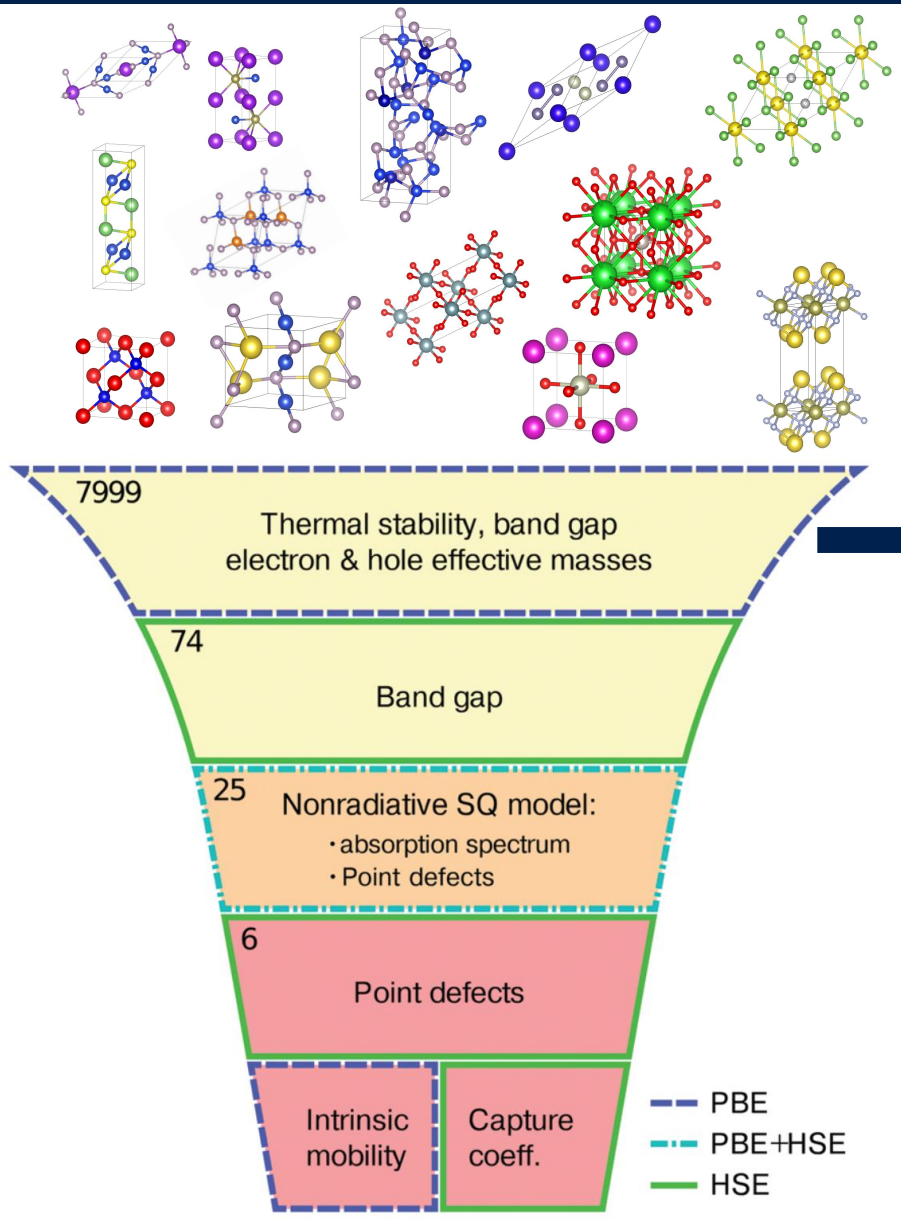
What can we do with that ? A few examples of what has been done



Looking for new materials for photovoltaics

"High-throughput computational search for high carrier lifetime, defect-tolerant solar absorbers", D. Dahliah et al., Energy & Environmental Science 14, 5057-5073 (2021)

What can we do with that ? A few examples of what has been done

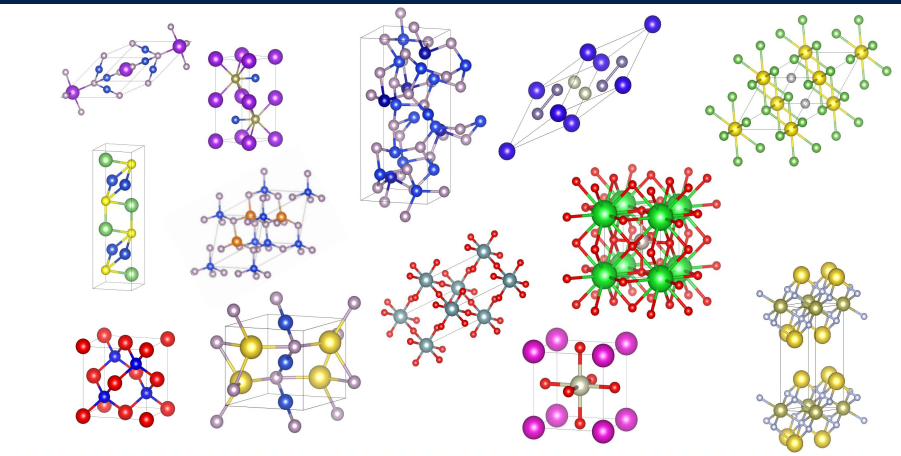


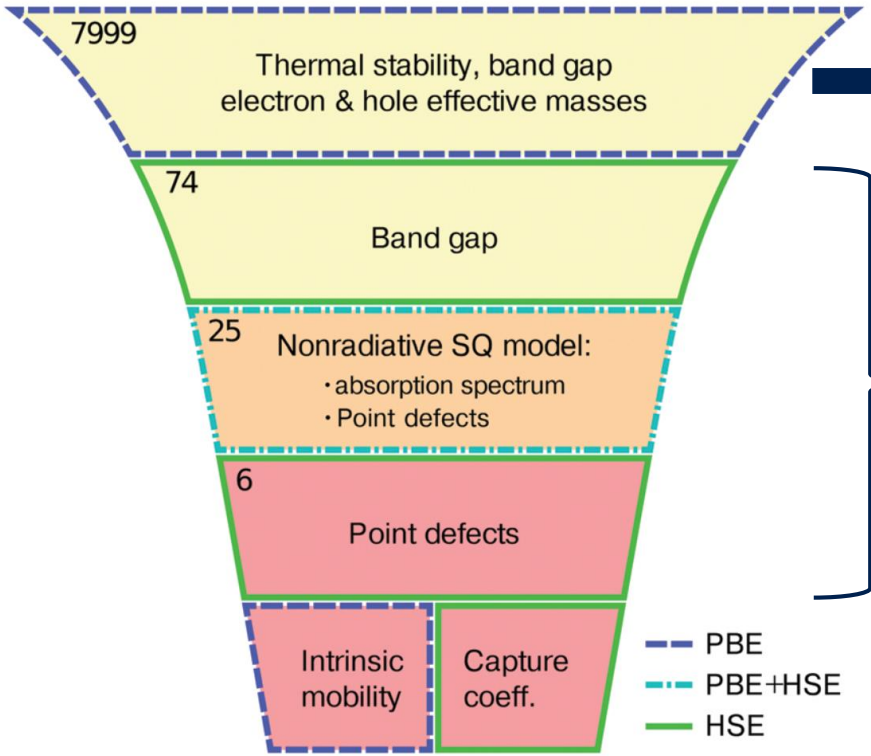
Looking for new materials for photovoltaics

Use of the previous database for this step

"High-throughput computational search for high carrier lifetime, defect-tolerant solar absorbers", D. Dahliah et al., Energy & Environmental Science 14, 5057-5073 (2021)

What can we do with that ? A few examples of what has been done





Looking for new materials for photovoltaics

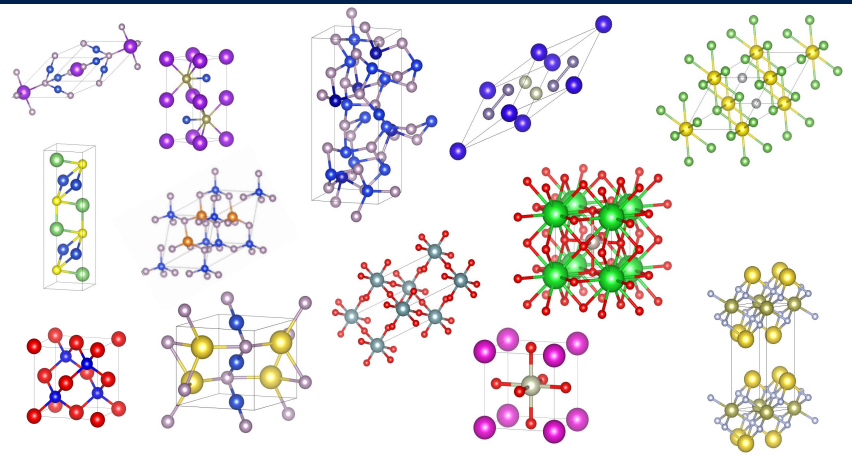
Use of the previous database for this step

atomate

FireWorks

"High-throughput computational search for high carrier lifetime, defect-tolerant solar absorbers", D. Dahliah et al., Energy & Environmental Science 14, 5057-5073 (2021)

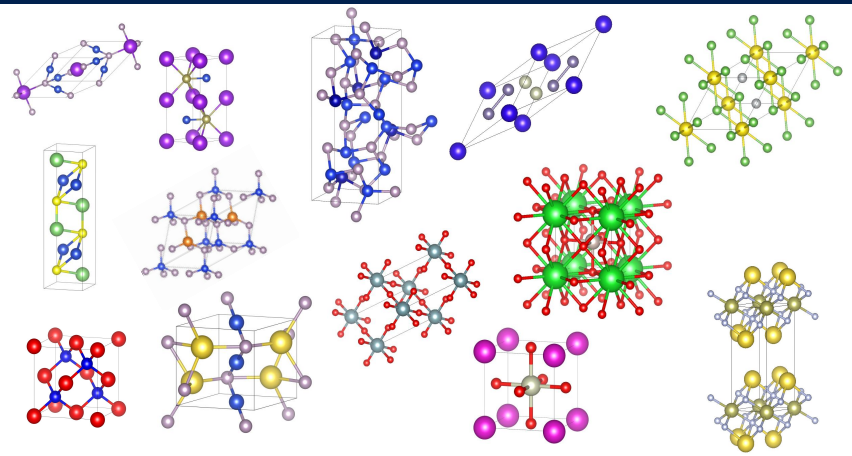
What can we do with that ? A few examples of what has been done



Computing the vibrational properties of materials (thermodynamics, transport,...)

"High-throughput density-functional perturbation theory phonons for inorganic materials", G. Petretto et al., Scientific Data 5, 180065 (2018)

What can we do with that ? A few examples of what has been done

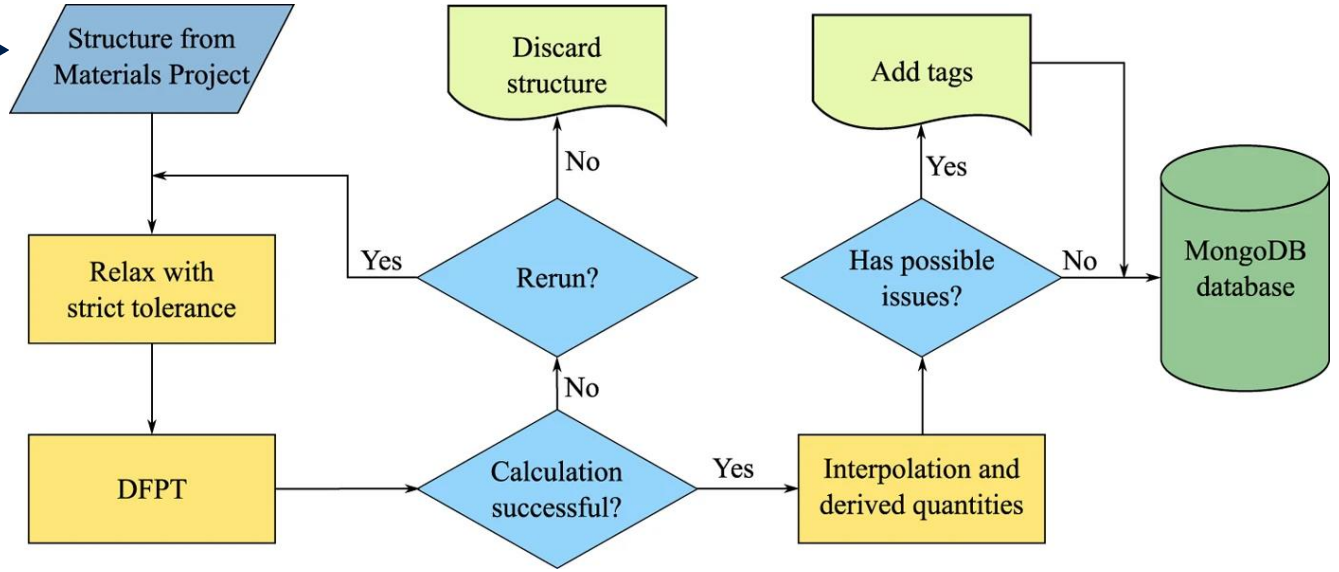


1521 materials

Computing the vibrational properties of materials (thermodynamics, transport,...)

abiflows

FireWorks



"High-throughput density-functional perturbation theory phonons for inorganic materials", G. Petretto et al., Scientific Data 5, 180065 (2018)

Take-home message

- Materials science sometimes require many computations
- FireWorks is a powerful tool to handle workflows
- It can be used in many applications : you should give it a try !

Thank you for your attention