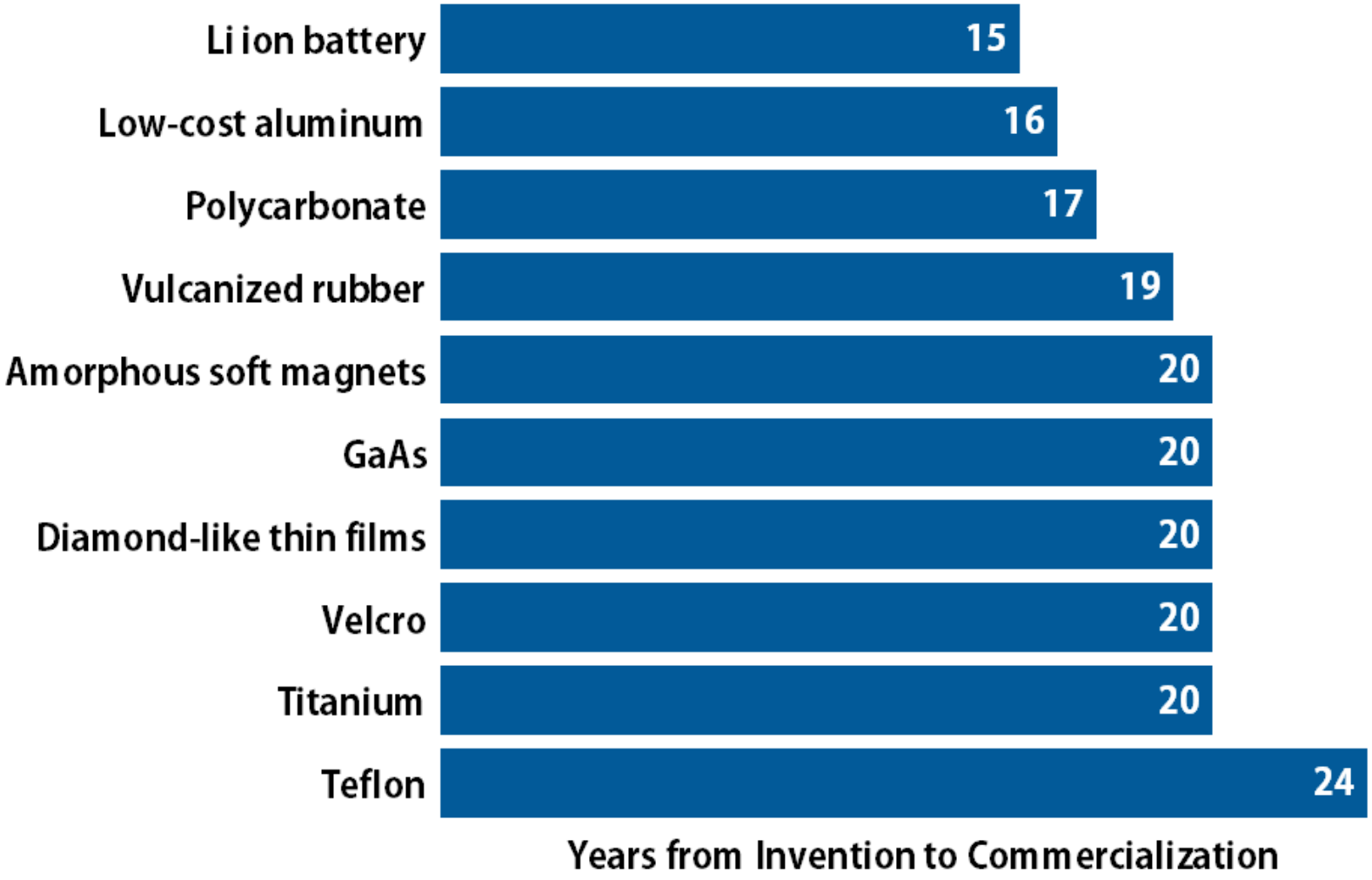
The Materials Project:

computing and sharing a searchable database of materials properties using the Fireworks job management system

Data Intensive Computing | June 2014

Anubhav Jain
Energy & Environmental Technologies
Berkeley Lab

It takes two decades to commercialize new materials



Materials data from: **Eagar T., King M.** *Technology Review 1995*

The lack of materials information slows materials innovation

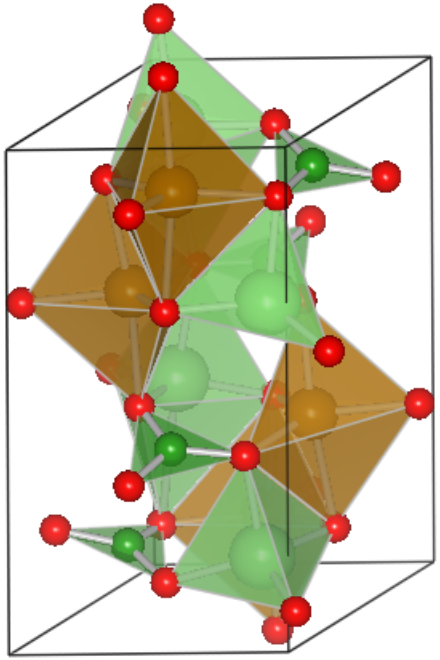
What are the properties of known materials?

What new, useful materials might exist?

How can I optimize a material over multiple criteria?

Often, 'experience' is the only guide.

The big idea: use computing + theory to predict materials properties on a large scale

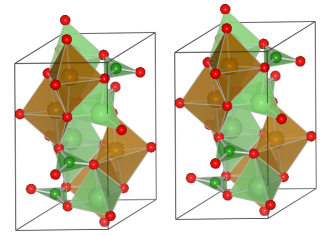


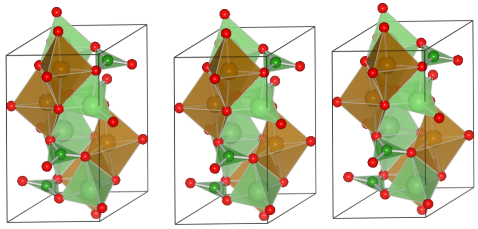
+

+

$$i\hbar \frac{d\Psi(\{r_i\};t)}{dt} = \hat{H}\,\Psi(\{r_i\};t)$$

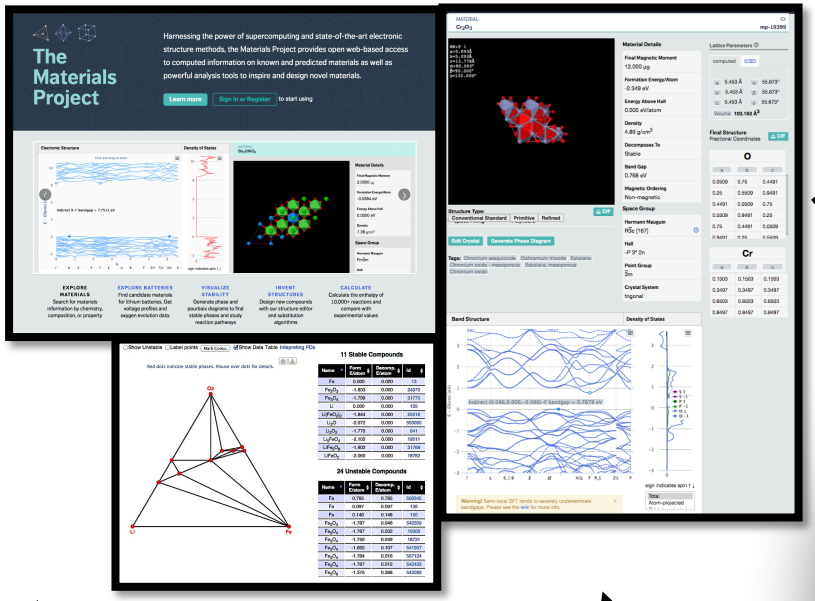
$$H = \sum_{i=1}^{N_e} \nabla_i^2 + \sum_{i=1}^{N_e} V_{nuclear}(r_i) + \sum_{i=1}^{N_e} V_{effective}(r_i)$$

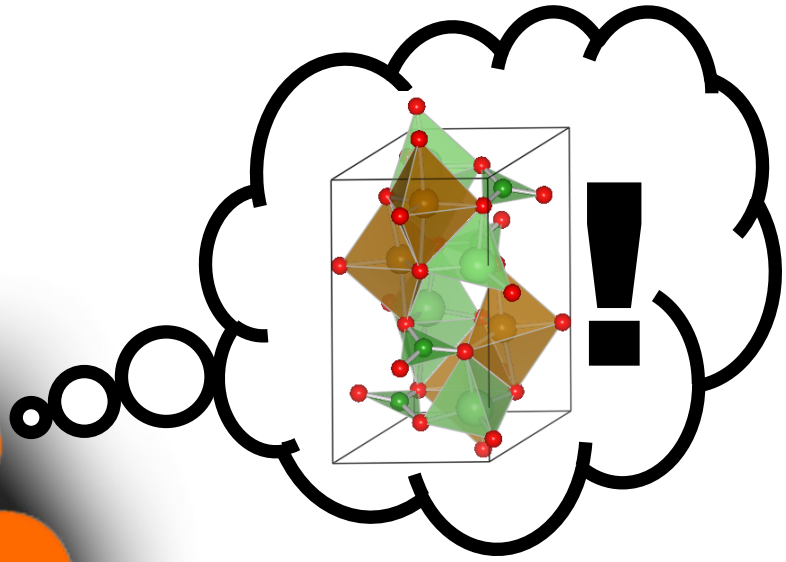


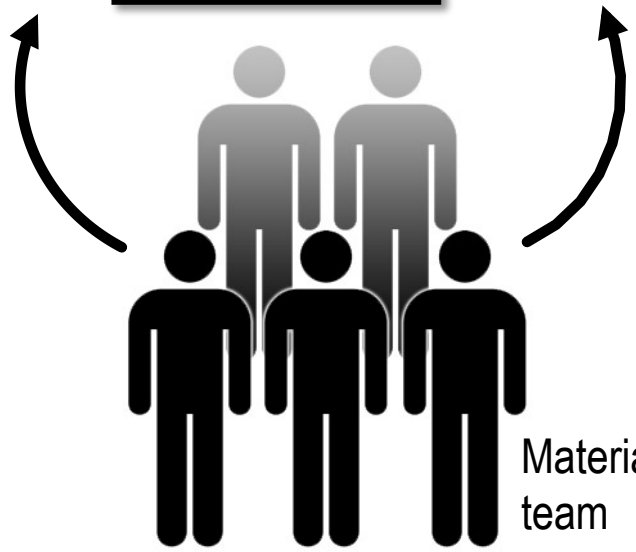


Total energy
Optimized structure
Magnetic ground state
Charge density
Band structure / DOS

The hope: unprecedented amounts of information will accelerate discoveries!







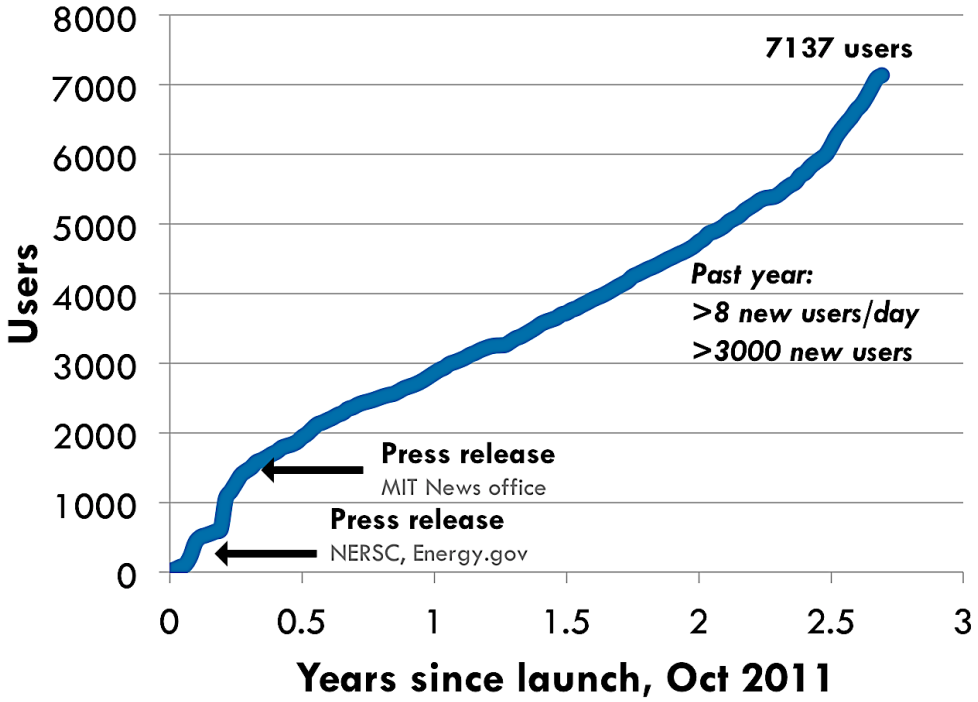
Materials Project team

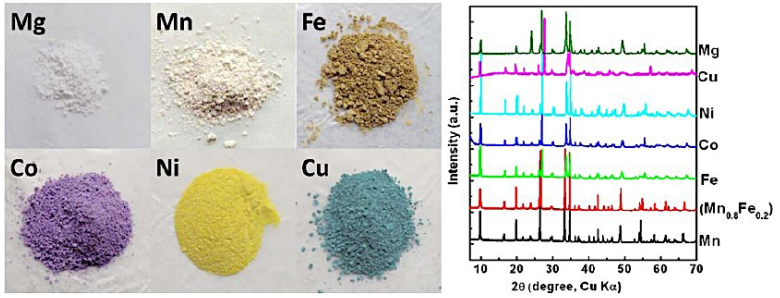
Any materials researcher

www.materialsproject.org – many milestones in under 3 years

- >7000 registered users
  - >1000 industry
- Internally – new battery materials
- >150 citations

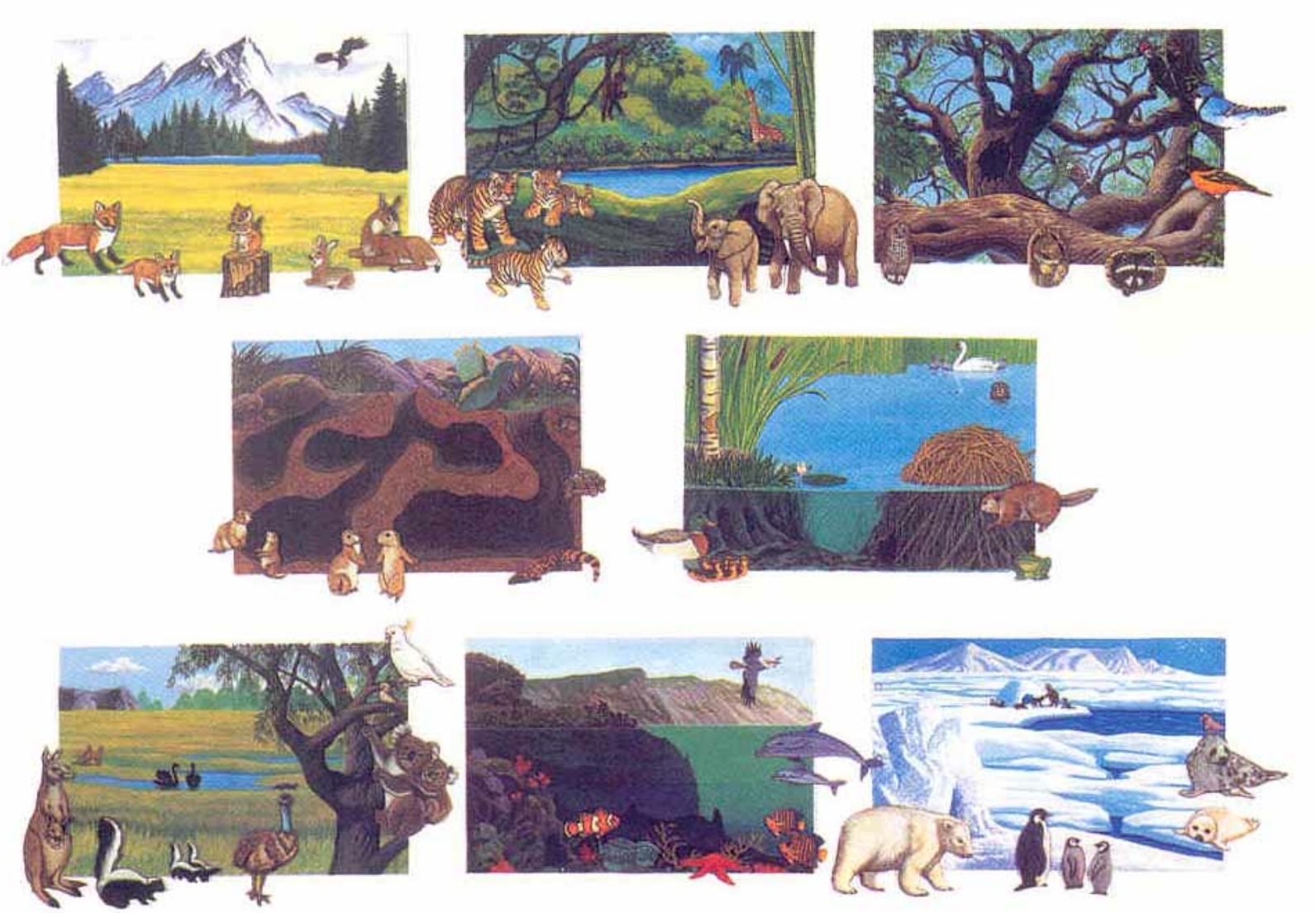
- 20 million CPU-hours
  - >50,000 materials
  - >300,000 DFT jobs
  - >1 million small jobs





$Li_3MPO_4CO_3$ cathode materials

Infrastructure – existing workflow software, or build our own?



http://www3.canisius.edu/~grandem/animalshabitats/animals.jpg

Need to keep track of millions of workflows over the course of several years

- Persistent record of jobs, not "run and forget"
- As new functionalities are available, new jobs are added depending on what was run before

FireWorks

Workflow Dashboard

738,489 Completed FireWorks as of 2014-06-16, 22:45 (PDT)

Recent Activity

Newest Fireworks

ID	Name	State
932696	C12_F12_Mn2_O8_P2_S1--VASP_db_insertion	FIZZLED
932695	C12_F12_Mn2_O8_P2_S1--GGA_Uniform_v2	COMPLETED
932694	C11_Fe2_O11_Ru2_Te2--VASP_db_insertion	FIZZLED
932693	C11_Fe2_O11_Ru2_Te2--GGA_band_structure_v2	COMPLETED
932692	F12_N6_Ni1_Ta2--VASP_db_insertion	FIZZLED
932691	F12_N6_Ni1_Ta2--GGA_static_v2	COMPLETED
932690	H4_Mn1_O9_S2--VASP_db_insertion	COMPLETED
932689	H4_Mn1_O9_S2--GGA_static_v2	COMPLETED
932688	Co1_N6_Np1_O8--VASP_db_insertion	COMPLETED
932687	Co1_N6_Np1_O8--GGA_static_v2	COMPLETED
932686	Cr13_Li3_Ni3_O96_S24--VASP_db_insertion	FIZZLED
932685	Cr13_Li3_Ni3_O96_S24--GGA_static_v2	COMPLETED
932684	Cl1_Fe1_Mo1_O4--VASP_db_insertion	COMPLETED
932683	Cl1_Fe1_Mo1_O4--GGA_static_v2	COMPLETED
932682	F4_H2_Mn1_O1_Rb1--VASP_db_insertion	COMPLETED
932681	F4_H2_Mn1_O1_Rb1--GGA_static_v2	COMPLETED
932680	H8_Mo1_N2_O2_S2--VASP_db_insertion	COMPLETED
932679	H8_Mo1_N2_O2_S2--GGA_static_v2	COMPLETED
932678	C1_Li3_Mn1_O7_P1--VASP_db_insertion	COMPLETED
932677	C1_Li3_Mn1_O7_P1--GGA_static_v2	COMPLETED

Current Database Status

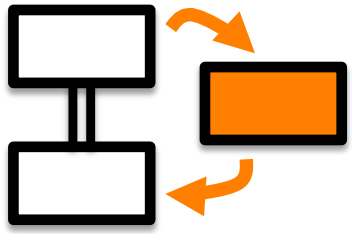
	Fireworks	Workflows
ARCHIVED	109,576	22,992
DEFUSED	5,581	2,705
WAITING	54,580	
READY	0	665
RESERVED	4,401	
FIZZLED	19,483	16,038
RUNNING	582	4,210
COMPLETED	738,489	115,363
TOTAL	932,692	161,974

Newest Workflows

ID	Name	State
705152	Cr2 O7 P2	COMPLETED
711108	Be3 Ca1 Mn2 O12 Si3	COMPLETED
678464	Ba2 I5 O17 V1	COMPLETED
503659	K8 Mo6 O24 Zr1	FIZZLED
558336	Fe2 Li1 O16 P5	COMPLETED
684673	C1 Li2 O7 P1 W1	COMPLETED
613141	H2 Mn3 O10 Se3	COMPLETED
704395	Fe3 Li3 O16 P4	COMPLETED
715652	Fe2 H3 O9 P3	COMPLETED
522886	Ca1 Mn4 O15 Si5	COMPLETED
680192	Li3 Mo3 O16 P4	COMPLETED
612610	C1 Cr1 Li2 O7 P1	COMPLETED
680213	Fe1 Li1 O7 P2	COMPLETED
685313	C1 Fe1 Li2 O7 P1	COMPLETED
932631	Li3 Mn3 O16 P4	COMPLETED
558979	Fe3 Li3 O16 P4	COMPLETED
711442	H2 Mn2 O6 S1	COMPLETED
534671	B2 Ca1 Mn1 O5	COMPLETED
692388	H4 Mn1 O9 S2	FIZZLED
526753	Al4 Mn2 O18 Si5	RUNNING

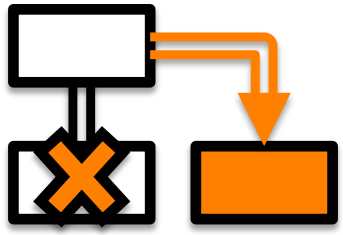
Very dynamic workflows

- Deal with failed calculations
- Results of initial calcs determine the remainder of the workflow in complex ways
- Rerun from mid-workflow essential



Detours

(about 10-20% of jobs fail and must be rerun with different input parameters)

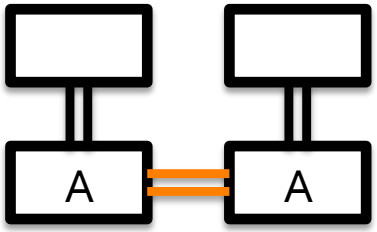


Branches

(based on the result of a calculation, the entire workflow might need to be modified)

Duplication a real problem

- Need to keep track of what was submitted
  - Which materials?
  - Which properties on those materials?
- Multiple users are submitting materials
- We need to make sure we don't duplicate work

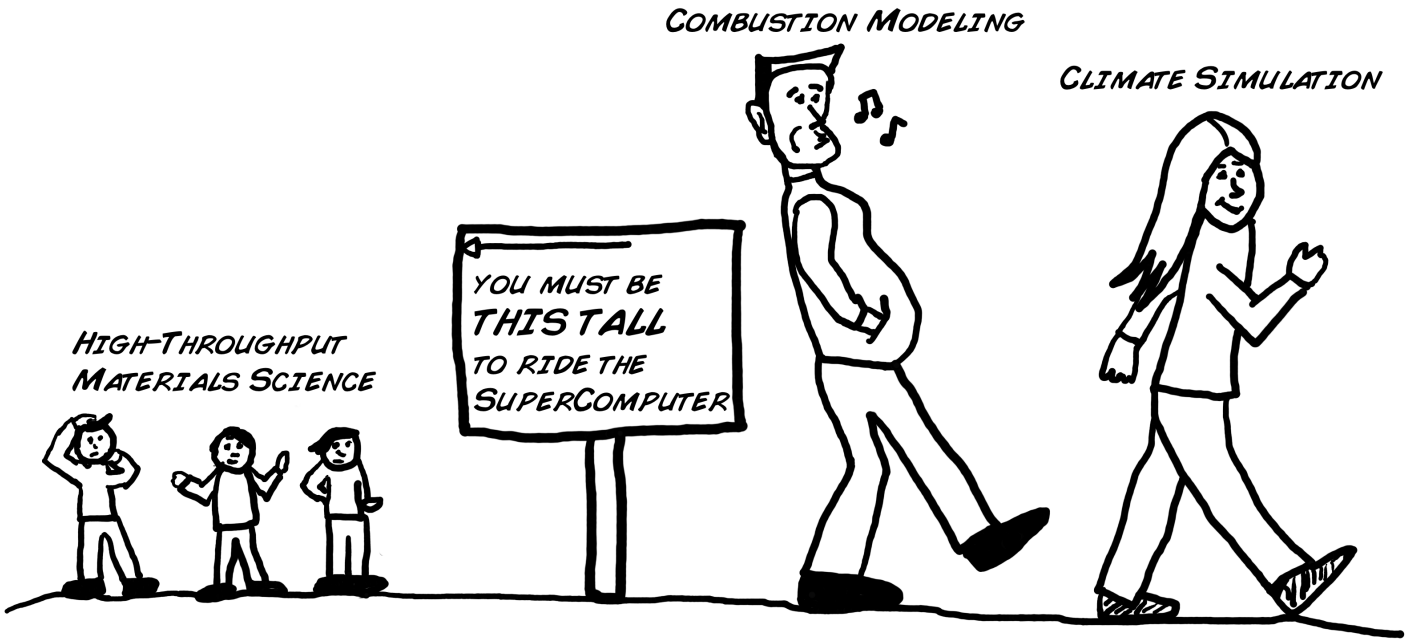


Duplicate Job detection

(if two workflows contain an identical step, ensure that the step is only run once and relevant information is still passed)

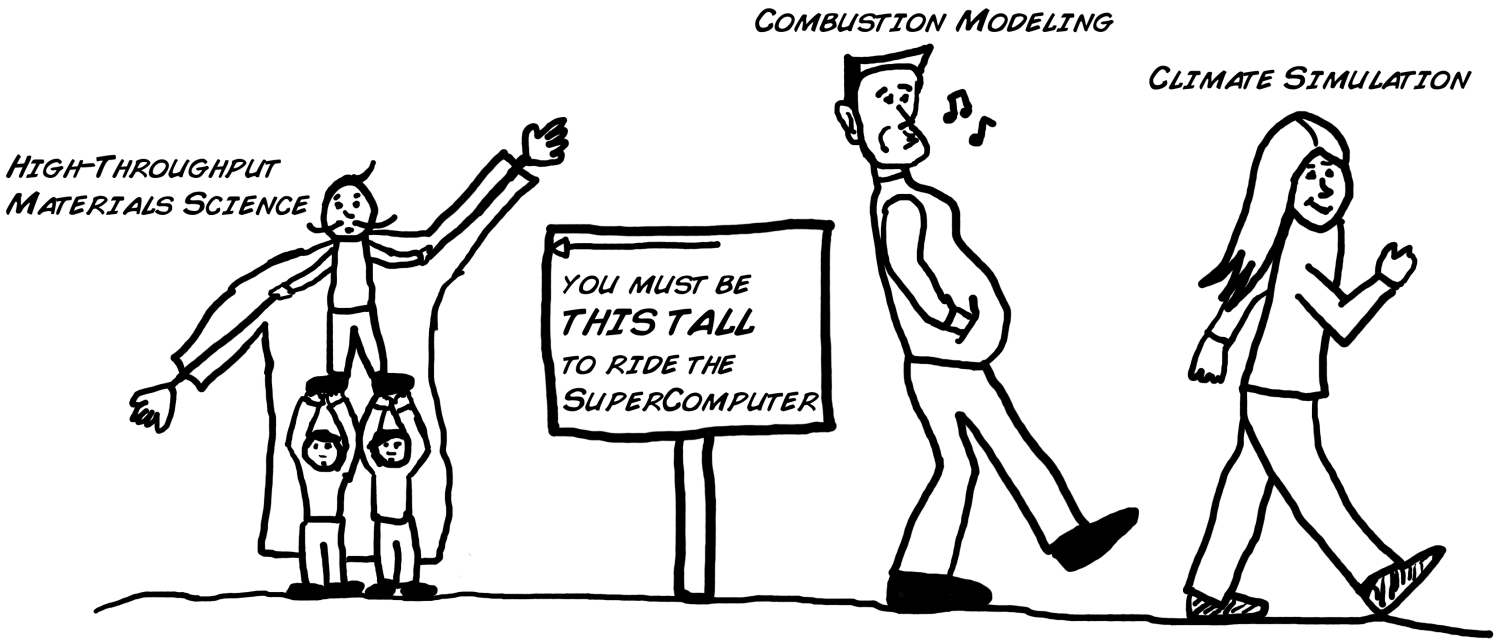
Support large computing centers

- Easy-to-install
  - FW currently at NERSC, SDSC, group clusters – Blue Gene planned
- Working within the limits of queue policies



Support large computing centers

- Easy-to-install
  - FW currently at NERSC, SDSC, group clusters – Blue Gene planned
- Working within the limits of queue policies



Docs + open source

- We wanted something we could get started with ourselves, without expert guidance or advice
- We wanted control of the source
  - e.g. FireWorks is on GitHub

A bird's eye view of FireWorks

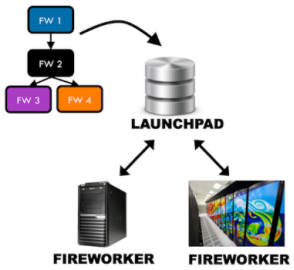
While FireWorks provides many features, its basic operation is simple. You can run FireWorks on a single laptop or at a supercomputing center.

Centralized Server and Worker Model

There are essentially just two components of a FireWorks installation:

- A **server** ("LaunchPad") that manages workflows. You can add workflows (a DAG of "FireWorks") to the LaunchPad, query for the state of your workflows, or rerun workflows. The workflows can be a straightforward series of scripts or dynamically adapt depending on the results obtained.
- One or more **workers** ("FireWorkers") that run your jobs. The FireWorkers request workflows from the LaunchPad, execute them, and send back information. The FireWorker can be as simple as the same workstation used to host the LaunchPad, or complicated like a national supercomputing center with a queueing system.

The basic infrastructure looks like this:



Add a Workflow

1. There are many ways to add Workflows to the database. You can do it directly from the command line as:

```
lpad add_scripts 'echo "hello"' 'echo "goodbye"' -n hello goodbye -w test_workflow
```

Output:

```
2013-10-03 13:51:19,991 INFO Added a workflow. id_map: {0: 1, 1: 2}
```

This added a two-job linear workflow. The first jobs prints *hello* to the command line, and the second job prints *goodbye*. We gave names (optional) to each step as "hello" and "goodbye". We named the workflow overall (optional) as "test_workflow".

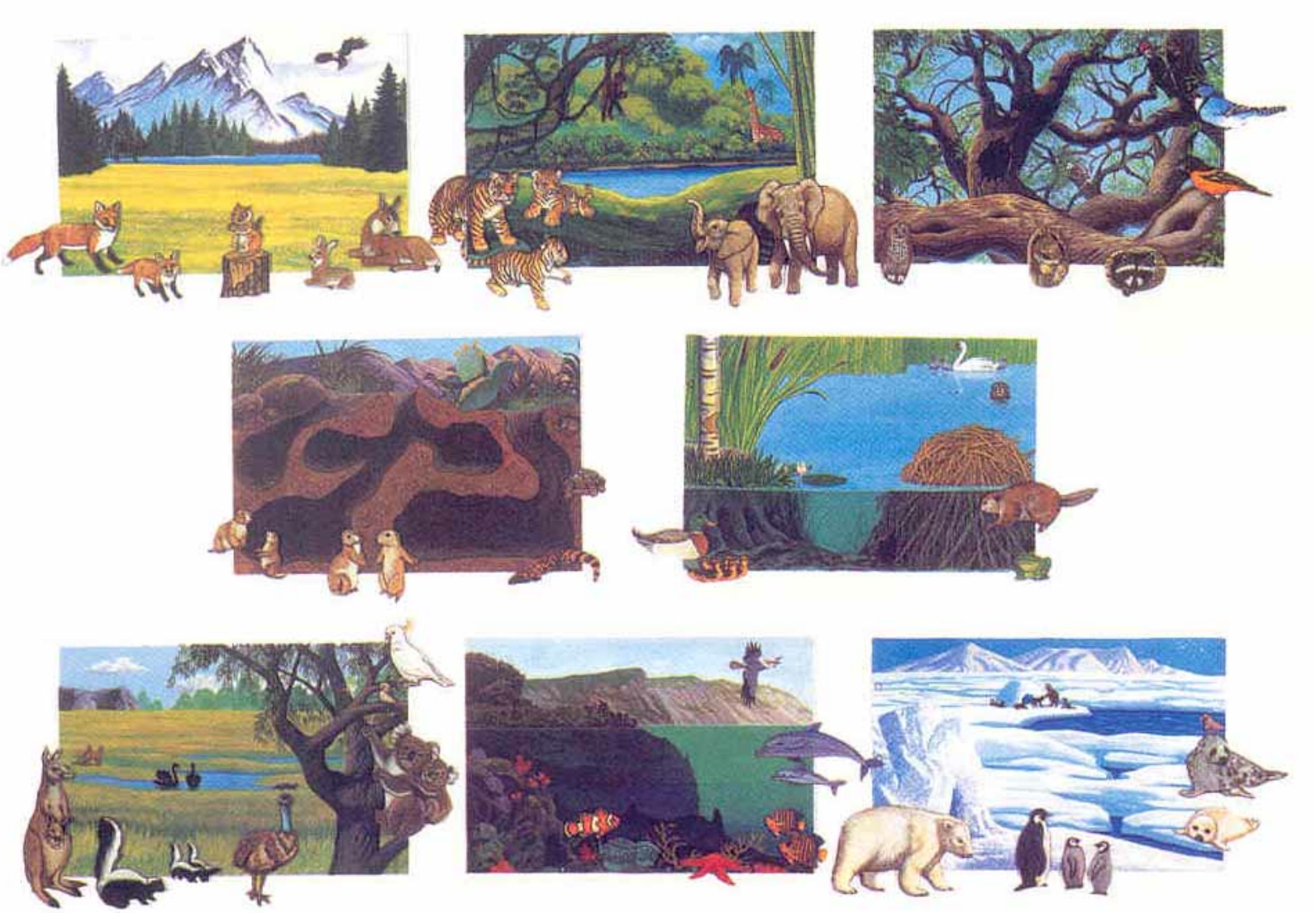
2. Let's look at our test workflow:

```
lpad get_wfs -n test_workflow -d more
```

Output:

```
{
    "name": "test_workflow",
    "state": "READY",
    "states": {
        "hello--1": "READY",
        "goodbye--2": "WAITING"
    },
    "created_on": "2014-02-10T22:10:27.024000",
    "launch_dirs": {
        "hello--1": [],
        "goodbye--2": []
    },
    "updated_on": "2014-02-10T22:10:27.029000"
}
```

We felt we had a new niche - that's why we added to the list of WF systems



http://www3.canisius.edu/~grandem/animalshabitats/animals.jpg

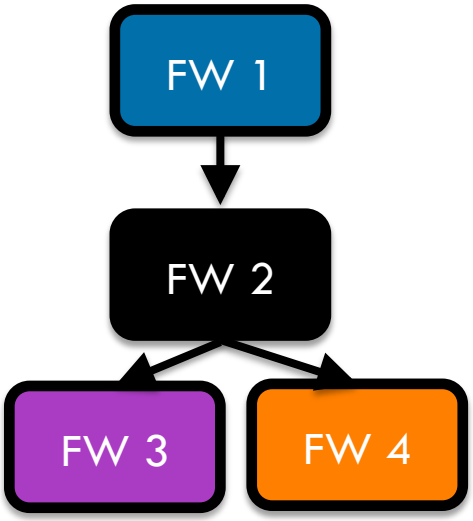
FireWorks workflow software project began ~1.5 years ago

The goal was not to take the workflow world by storm, but to build something for our needs that was still **very general** and also **extensible by the community**

Basic FireWorks operation

ROCKET LAUNCHER / QUEUE LAUNCHER

Directory 1 | Directory 2



LAUNCHPAD

Workflows are simple JSON/YAML documents that have very little "fluff"

(this is YAML, a bit prettier for humans
but less pretty for computers)

```
fws:
- fw_id: 1
  spec:
    _tasks:
    - _fw_name: ScriptTask:
      script: echo 'To be, or not to be,'
- fw_id: 2
  spec:
    _tasks:
    - _fw_name: ScriptTask
      script: echo 'that is the question:'
links:
  1:
  - 2
metadata: {}
```

The same JSON document will produce the same result on any computer (with the same Python functions).

JSON + MongoDB means you can store workflows directly and make rich queries

(this is YAML, a bit prettier for humans but less pretty for computers)

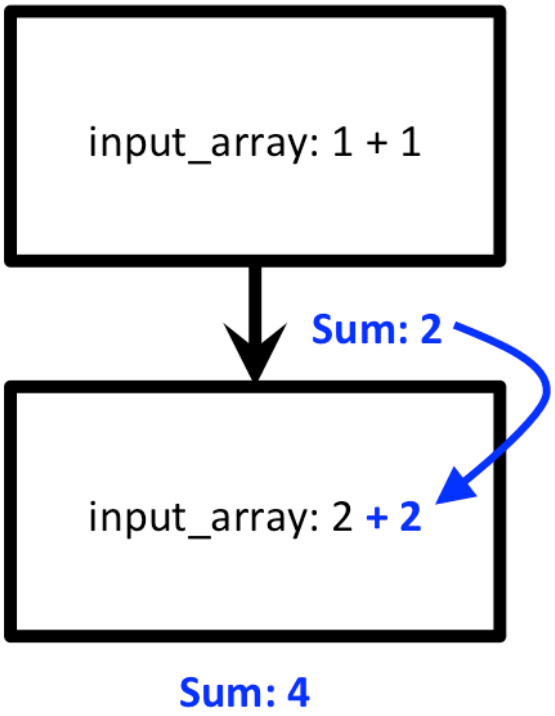
```
fws:
- fw_id: 1
  spec:
    _tasks:
    - _fw_name: ScriptTask:
      script: echo 'To be, or not to be,’
- fw_id: 2
  spec:
    _tasks:
    - _fw_name: ScriptTask
      script: echo 'that is the question:’
links:
  1:
  - 2
metadata: {}
```

Just some of your search options:

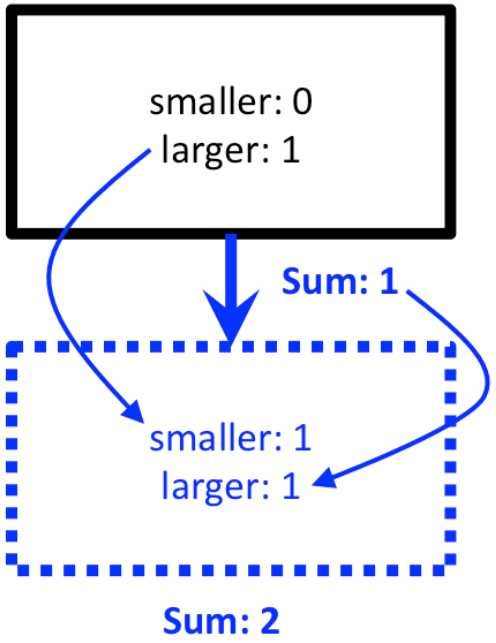
- simple matches
- match in array
- greater than/less than
- regular expressions
- match subdocument
- Javascript function
- MapReduce…

All for free, and all on the native workflow format!

Jobs can return objects that modify workflows or future jobs via JSON language



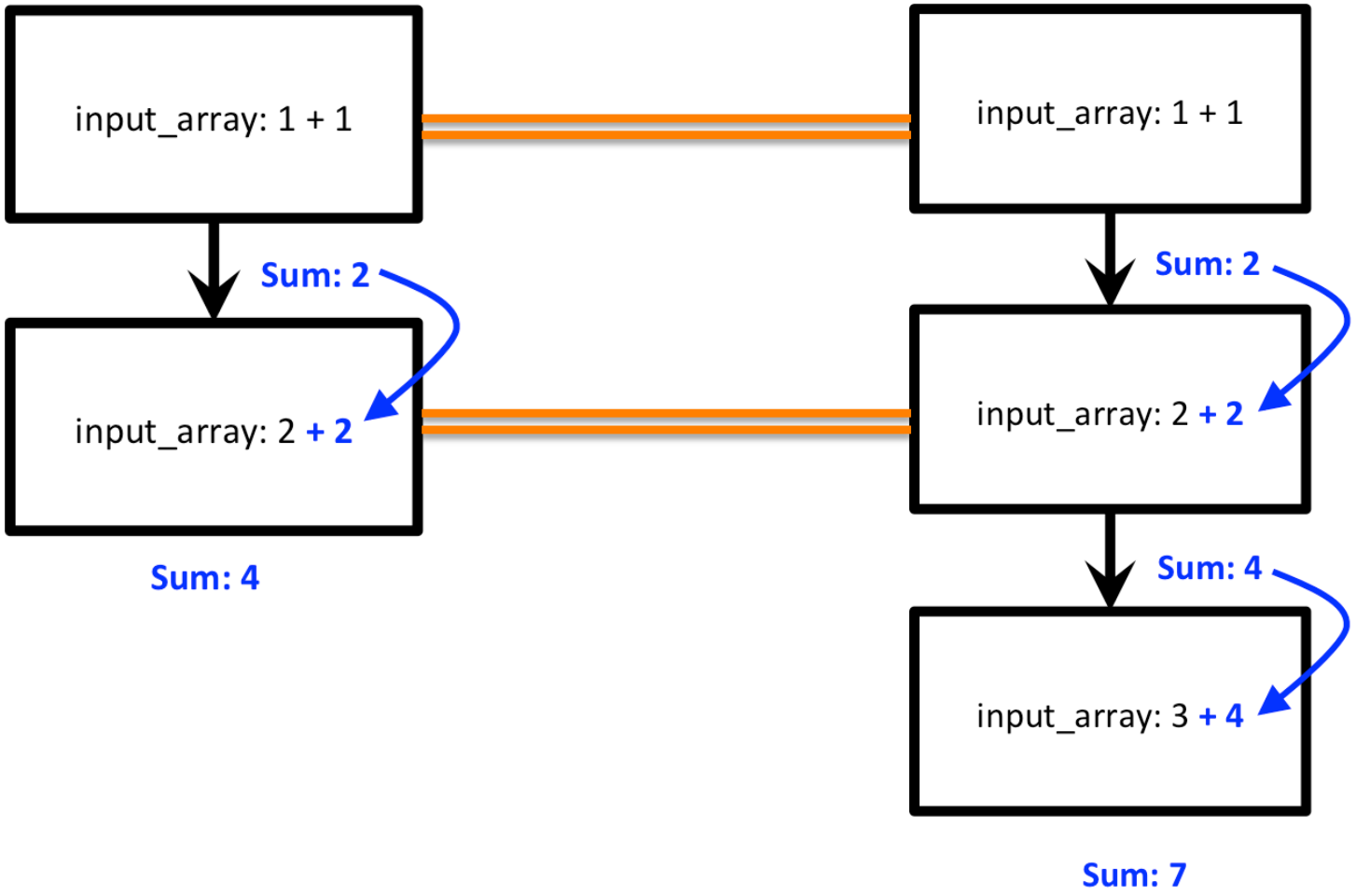
Use MongoDB's dictionary update language to allow for JSON document updates



Workflows can create new workflows or add to current workflow

- a recursive workflow
- calculation "detours"
- branches

JSON duplicate checking simple and automatic



Many execution modes

- Run directly on a node
- Run a single job within a PBS script
  - generic or highly tailored to the job, e.g. walltime
- Consecutively pull many jobs in a PBS script
- Run via PBS/SGE/SLURM/NEWT
- Distribute jobs over many workers/computers
- Pack jobs and pretend to be a big job without any setup

How does Materials Project use FireWorks?

- As the main workhorse for managing DFT computations
- For "computations on demand" of smaller structure prediction jobs
  - duplicate check essential!
- As part of a "mission simulator" for developing new features

FireWorks is free and open-source

http://pythonhosted.org/FireWorks/

or Google "Python FireWorks"

(or maybe even "Python workflow software")