

Workflow Management with Fireworks Using FilePad

Johannes Hörmann

Department of Microsystems Engineering (IMTEK)

University of Freiburg, Germany

johannes.hoermann@imtek.uni-freiburg.de

1. **Sample use case 1: Gold droplet hits substrate**
2. **Sample use case 2: Parametric study**

1. **Sample use case 1: Gold droplet hits substrate**
2. **Sample use case 2: Parametric study**

Sample Case 1: Gold Cluster



Tasks

In this task we will melt a gold *cluster*, i.e. a finite sized object of just gold atoms, and compute its heat capacity. Cluster, such as the one pictured, are used in applications ranging from catalysis to drug delivery.

...

Task 2: These clusters are cut from an ideal crystal, they need to be relaxed first. Relax them (minimize the potential energy).

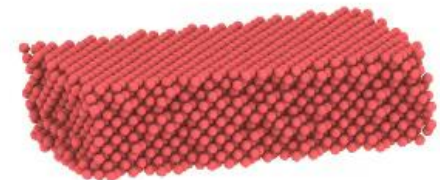
Task 3: Equilibrate the cluster at a low temperature, e.g. 100 K.

...

Task 5: Stepwise increase the temperature of the cluster until it melts. You will need to leave time for the cluster to equilibrate after the temperature changes. Use the `fix temp/rescale` command.

...

Extra task: Shoot the molten cluster onto a (colder) crystalline substrate.



Source: Lars Pastewka's MD tutorials

Sample Case 1: Gold Cluster

FilePad (python snippet)

```
from fireworks import LaunchPad, Firework, Workflow

lp = LaunchPad.auto_load()

# direct FilePad access, similar to the familiar LaunchPad:
from fireworks.utilities.filepad import FilePad

fp = FilePad.auto_load()
```

Sample Case 1: Gold Cluster

Set of input and data files (python snippet)

```
# input and data files for the current parametric study

files = {
    'potential': 'Au-Grochola-JCP05.eam.alloy',
    'large_cluster': 'cluster_3871.lammps',
    'small_cluster': 'cluster_923.lammps',
    'minimize_substrate': 'A_minimize_substrate.in',
    # ...
}
```

Sample Case 1: Gold Cluster

Set of input and data files (python snippet)

```
project_id = 'fireworks-hands-on-2019-12-09'

# insert these input files into FilePad
for name, file in files.items():
    identifier = '/'.join((project_id,name))
    fp_files.append(
        fp.add_file(
            file, identifier=identifier,
            metadata = {
                'project': project_id,
                'type': 'input',
                'name': name
            }
        )
    )
```

Sample Case 1: Gold Cluster

GridFS ID and identifier

```
pprint(fp_files)
```

```
>> [  
... ('5ded279c11940f1ec729018a', 'fireworks-hands-on-2019-12-09/potential'),  
... ('5ded279c11940f1ec729018d', 'fireworks-hands-on-2019-12-09/large_cluster'),  
... ('5ded279c11940f1ec7290190', 'fireworks-hands-on-2019-12-09/small_cluster'),  
... ('5ded279c11940f1ec7290193', 'fireworks-hands-on-2019-12-09/minimize_substrate'),  
... #...  
... ]
```


Sample Case 1: Gold Cluster

Query files in FilePad:

```
query = {  
    'metadata.project': project_id,  
    'metadata.type': 'input'  
}
```

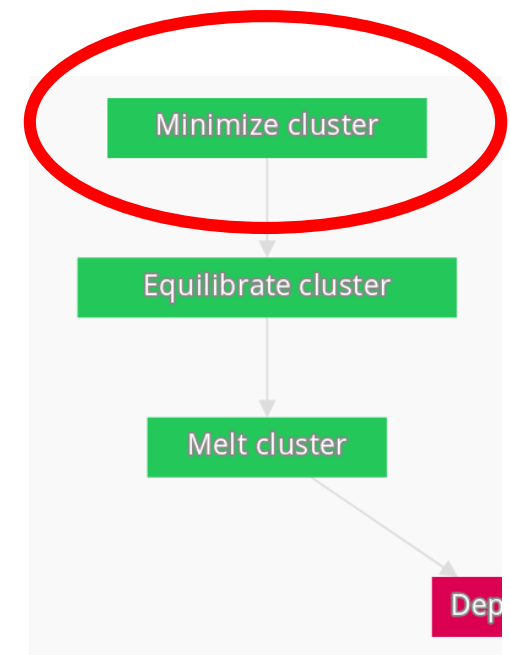
```
fp.filepad.count_documents(query)
```

```
>> 9
```

Sample Case 1: Gold Cluster

```
fw_id: -10
name: "Minimize substrate"
spec:
  _category: "bwcloud_noqueue"
  _files_out:
    data_file: final.lammps
  tasks:
    - _fw_name: GetFilesTask
      identifiers:
        - fireworks-hands-on-2019-12-09/potential
      new_file_names:
        - Au-Grochola-JCP05.eam.alloy
    - _fw_name: GetFilesTask
      identifiers:
        - fireworks-hands-on-2019-12-09/minimize_substrate
      new_file_names:
        - lammps.in
    - _fw_name: ScriptTask
      script: source /etc/profile; module load LAMMPS; mpirun -n 2 lmp -in lammps.in
```

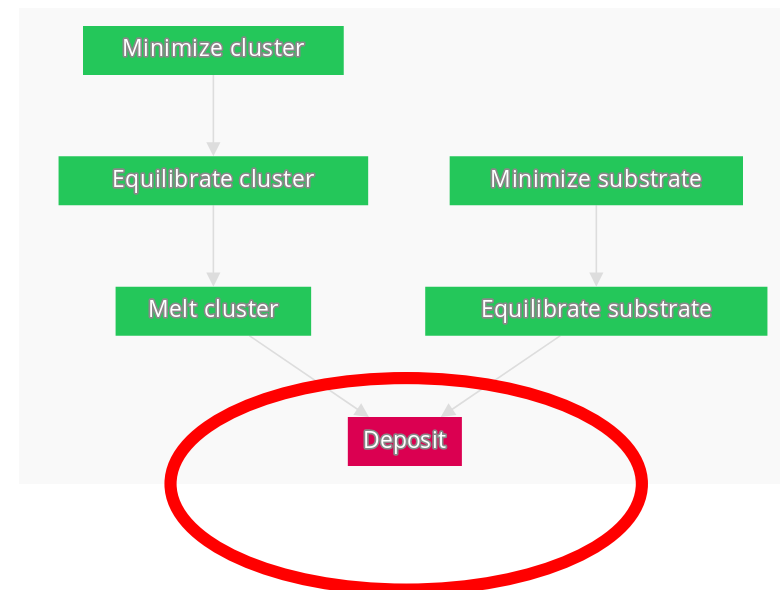
Minimize substrate:
A_substrate.yaml



Sample Case 1: Gold Cluster

```
- _fw_name: AddFilesTask
  compress: true
  metadata:
    cluster: large_cluster
    cluster_size: 3871
    material: gold
    project: fireworks-hands-on-2019-12-09
    type: trajectory
    velocity: -50
  paths: traj.dump
```

Deposit cluster on substrate:
Excerpt from D_deposit.yaml



Sample Case 1: Gold Cluster

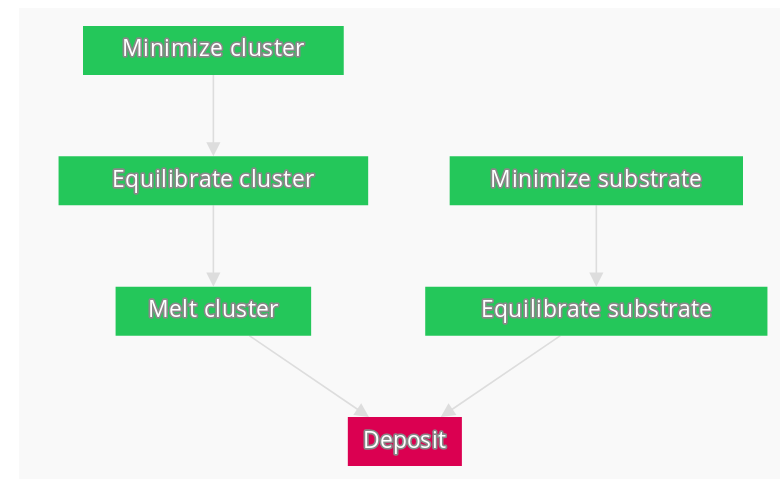
```
# We have each FireWork prepared in a static yaml file within this folder.  
# Now, we read these and fix the dependencies here:
```

```
fw_A_substrate = Firework.from_file('A_substrate.yaml',)  
fw_B_substrate = Firework.from_file('B_substrate.yaml')
```

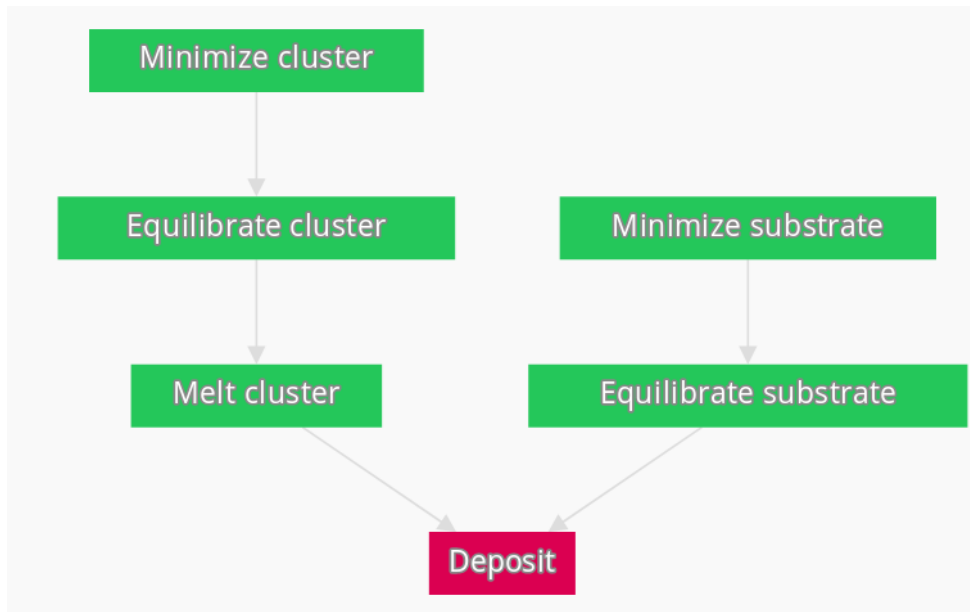
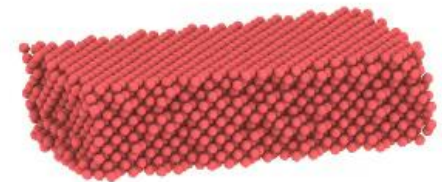
```
fw_A_cluster   = Firework.from_file('A_cluster.yaml')  
fw_B_cluster   = Firework.from_file('B_cluster.yaml')  
fw_C_cluster   = Firework.from_file('C_cluster.yaml')
```

```
fw_D_deposit   = Firework.from_file('D_deposit.yaml')
```

```
wf = Workflow(  
    [ fw_A_substrate, fw_B_substrate,  
      fw_A_cluster, fw_B_cluster, fw_C_cluster,  
      fw_D_deposit ],  
  
    { fw_A_substrate: fw_B_substrate,  
      fw_B_substrate: fw_D_deposit,  
  
      fw_A_cluster:   fw_B_cluster,  
      fw_B_cluster:   fw_C_cluster,  
      fw_C_cluster:   fw_D_deposit },  
  
    name = "Gold cluster impact")  
  
fw_ids = lp.add_wf(wf)
```



Sample Case 1: Gold Cluster



Sample Case 1: Gold Cluster

Query results:

```
query = {  
    "metadata.project": project_id,  
    "metadata.material": 'gold',  
    "metadata.type": 'trajectory'  
}
```

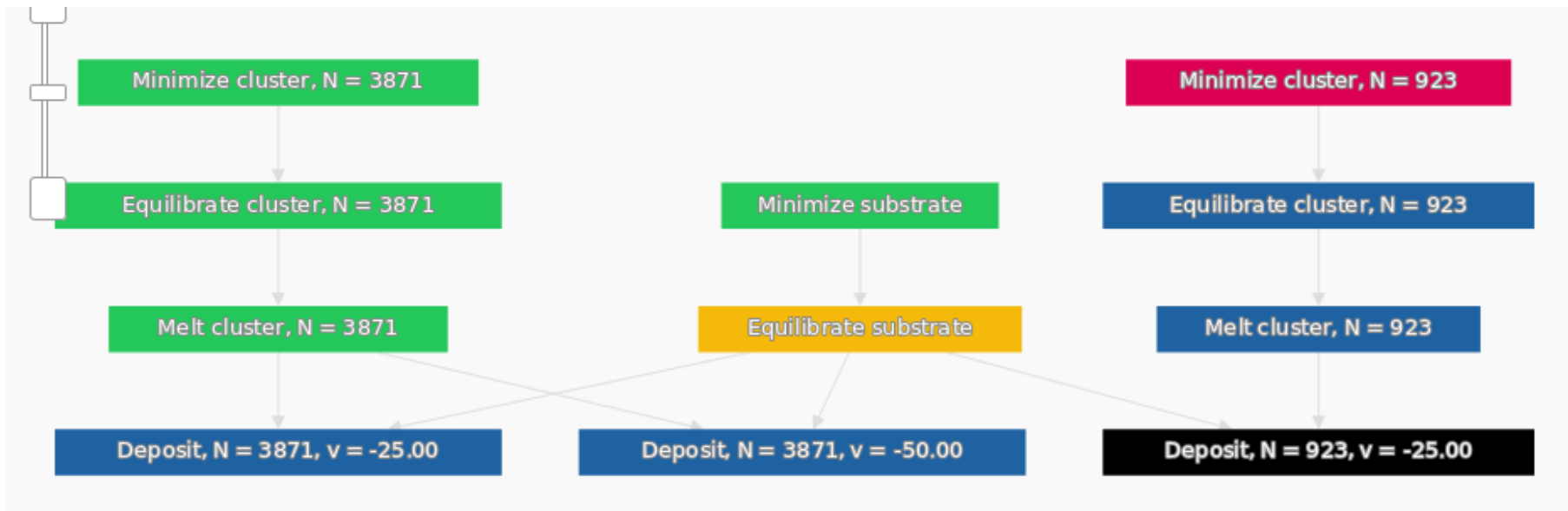
```
traj_file, traj_metadata = fp.get_file_by_query(query)[0]
```

1. **Sample use case 1: Gold droplet hits substrate**
2. **Sample use case 2: Parametric study**

Sample Case 2: Parametric study

Two parametric dimensions: cluster size and velocity

velocity	-50	-25
cluster_size		
923	False	True
3871	True	True



Sample Case 2: Parametric Study

Query files in FilePad:

```
query = {  
    'metadata.project': project_id,  
    'metadata.type': 'trajectory'  
}
```

```
fp.filepad.count_documents(query)
```

```
>> 3
```

Sample Case 2: Parametric Study

```
# first filter all files by a general query, here as above
# all trajectories in project
match_aggregation = {
    "$match": query
}
```

```
In [57]: # group by parameters of interest, i.e. cluster_size and velocity:
group_aggregation = {
    "$group": {
        "_id": {
            "cluster_size": "$metadata.cluster_size",
            "velocity": "$metadata.velocity"
        }
    }
}
```

```
In [58]: aggregation_pipeline = [ match_aggregation, group_aggregation ]
```

```
In [59]: # this aggregation yields all unique parameter sets as documents
cursor = fp.filepad.aggregate(aggregation_pipeline)
```

```
In [60]: # we sort these sets by cluster_size
unique_parameter_sets = [ c["_id"] for c in sorted(
    cursor, key=lambda d: d["_id"]["cluster_size"]) ]
```

Thereby, we get a simple overview on the unique combinations of parameters "cluster_size" and "velocity" that reside in our database:

```
In [61]: unique_parameter_sets
```

```
Out[61]: [{'cluster_size': 923, 'velocity': -25},
          {'cluster_size': 3871, 'velocity': -25},
          {'cluster_size': 3871, 'velocity': -50}]
```

Sample Case 2: Parametric Study

“slice” results at fixed cluster size...

```
query = {  
    "metadata.project":    project_id,  
    "metadata.material":   "gold",  
    "metadata.type":       "log",  
    "metadata.cluster_size": 3871}
```

... or at fixed velocity...

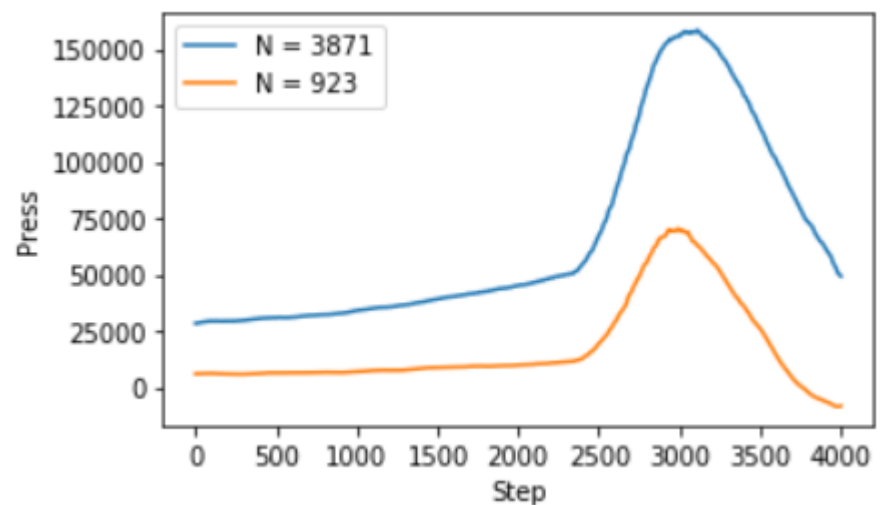
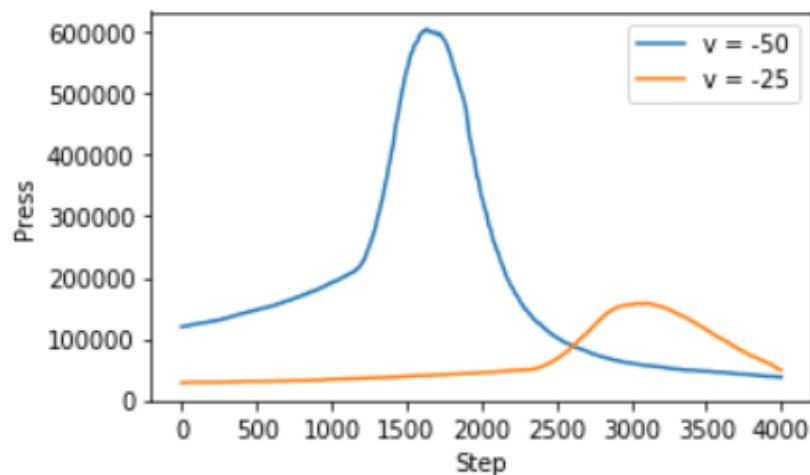
```
query = {  
    "metadata.project":    project_id,  
    "metadata.material":   "gold",  
    "metadata.type":       "log",  
    "metadata.velocity":   -25}
```

... and easily compare along parametric dimension ...

```
print(fp.filepad.count_documents(query) )  
# that many documents matching query
```

```
: files = fp.get_file_by_query(query)
```

2



Proper metadata facilitates meaningful evaluation!

Same principles apply to “FireWorks” and workflows themselves, not only to “files”!

Thank you!