Molecular simulation control and extension with gmxapi for GROMACS

BioExcel Webinar Series

Presenter: Eric Irrgang, *University of Virginia*
Host: Rossen Apostolov

19th September, 2018

Partners

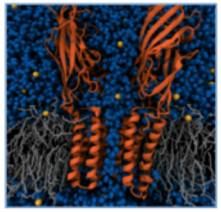
Funding

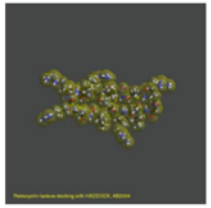
Horizon 2020
European Union Funding
for Research & Innovation

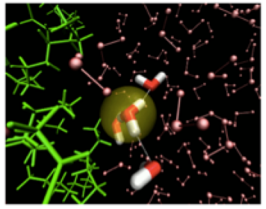
This webinar is being recorded

BioExcel Overview

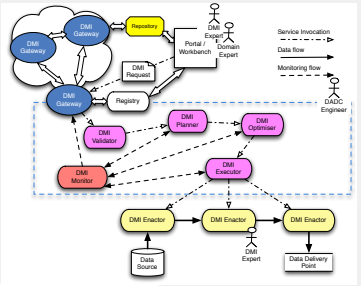
- **Excellence in Biomolecular Software**
  - Improve the performance, efficiency and scalability of key codes


MD simulations
/GROMACS/


Docking
/HADDOCK/


QM/MM
/CPMD/

- **Excellence in Usability**
  - Devise efficient workflow environments with associated data integration



Key Workflows and Platforms

Galaxy PROJECT · Open PHACTS · COMPSs · Open for Innovation KNIME · Apache Taverna

- **Excellence in Consultancy and Training**
  - Promote best practices and train end users

Interest Groups

- Integrative Modeling IG
- Free Energy Calculations IG
- Hybrid methods for biomolecular systems IG
- Biomolecular simulations entry level users IG
- Practical applications for industry IG
- Training IG
- Workflows IG

Support platforms

http://bioexcel.eu/contact

 **Forums**

 **Code Repositories**

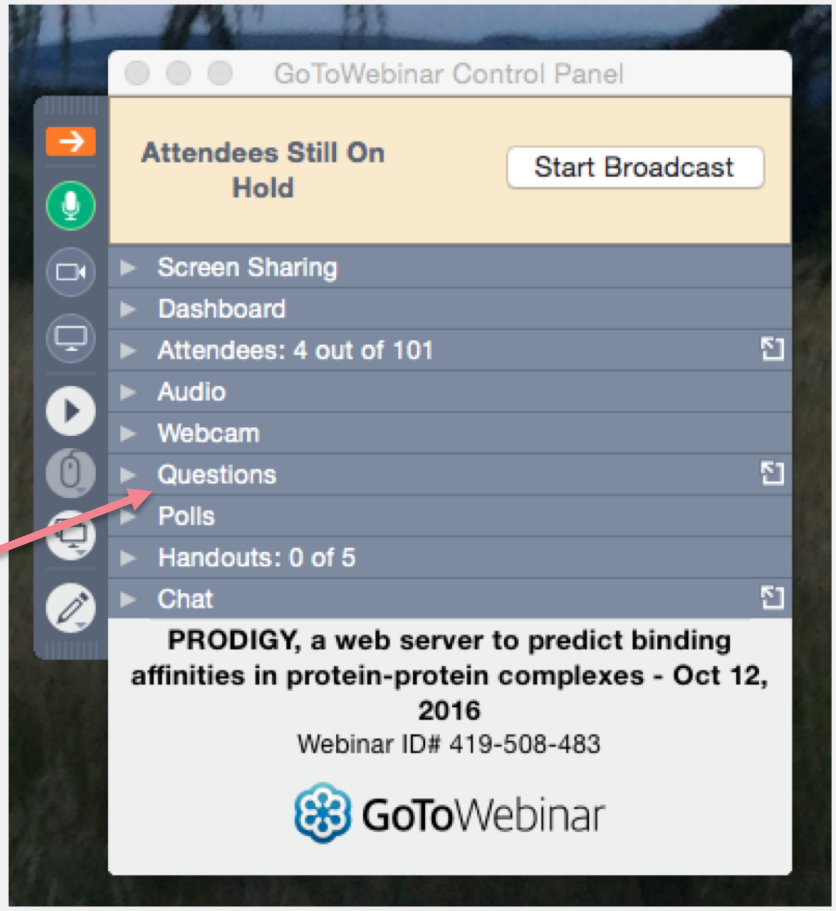
 **Chat Channel**

 **Video Channel**

Audience Q&A session

Please use the **Questions** function in GoToWebinar application

Any other questions or points to discuss after the live webinar? Join the discussion the discussion at http://ask.bioexcel.eu.



Today's Presenter

Eric Irrgang, University of Virginia

ericirrgang@gmail.com

Eric completed his undergraduate degree at the University of Texas, Austin and his PhD in Materials Science & Engineering with Sharon Glotzer at the University of Michigan before joining the Kasson Lab as a postdoctoral fellow. Now he is building interfaces for flexible and extensible molecular dynamics simulation. Eric believes strongly in proper software engineering design and flexible simulation interfaces but keeps a soft spot in his heart for Monte Carlo methods. He is supported by a MolSSI Software Fellowship.

Molecular simulation control and extension with gmxapi for GROMACS

M. Eric Irrgang
Jennifer M. Hays
Peter M. Kasson

BioExcel webinar series
19 September, 2018

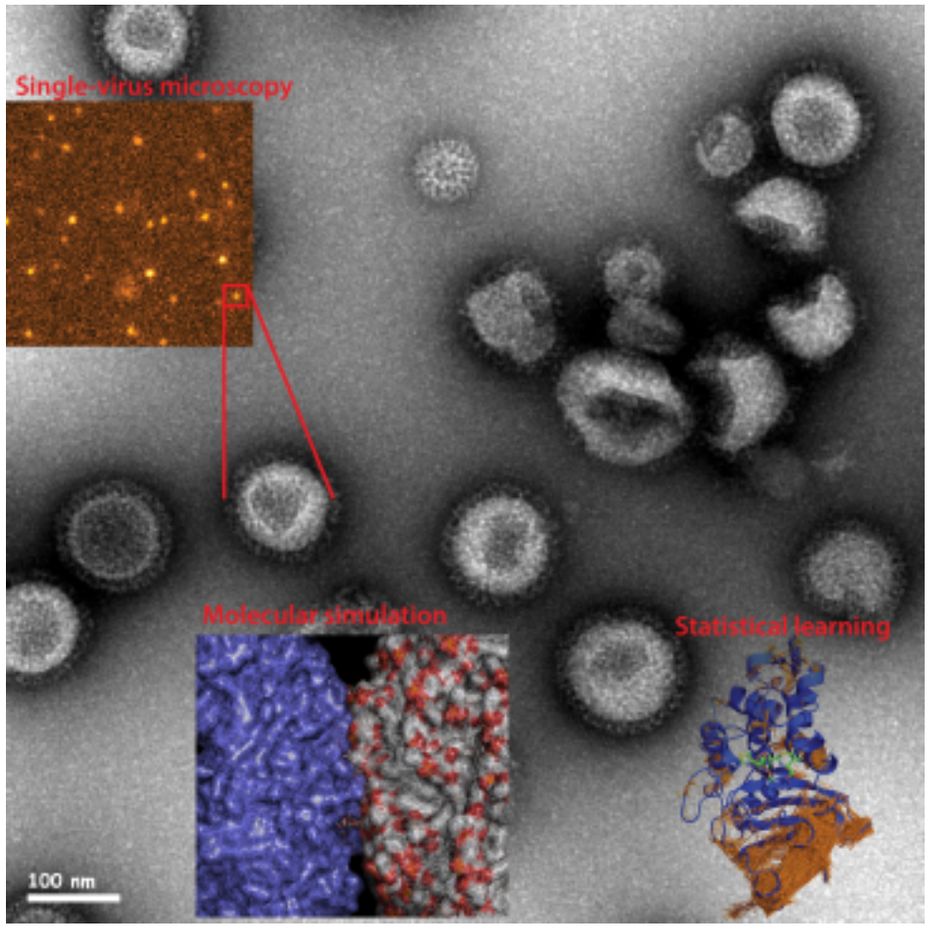
Kasson lab

research on physical mechanisms in infectious disease

- infection by enveloped viruses
- drug-resistant bacteria

tools and methods development

- combining experiments with advanced computation
- large-scale biomolecular simulation
- statistical learning



Simple Python interface, C++ performance

```
>>> md = gmx.from_file([filename1, filename2, filename3, ...])
```

```
>>> potential = myplugin.EnsembleRestraint(sites, *args, **kwargs)
```

```
>>> md.add_dependancy(potential)
```

```
>>> gmx.run()
```

Bioinformatics, 2018, 1–3
doi: 10.1093/bioinformatics/bty484
Advance Access Publication Date: 15 June 2018
Applications Note

Structural bioinformatics

gmxapi: a high-level interface for advanced control and extension of molecular dynamics simulations

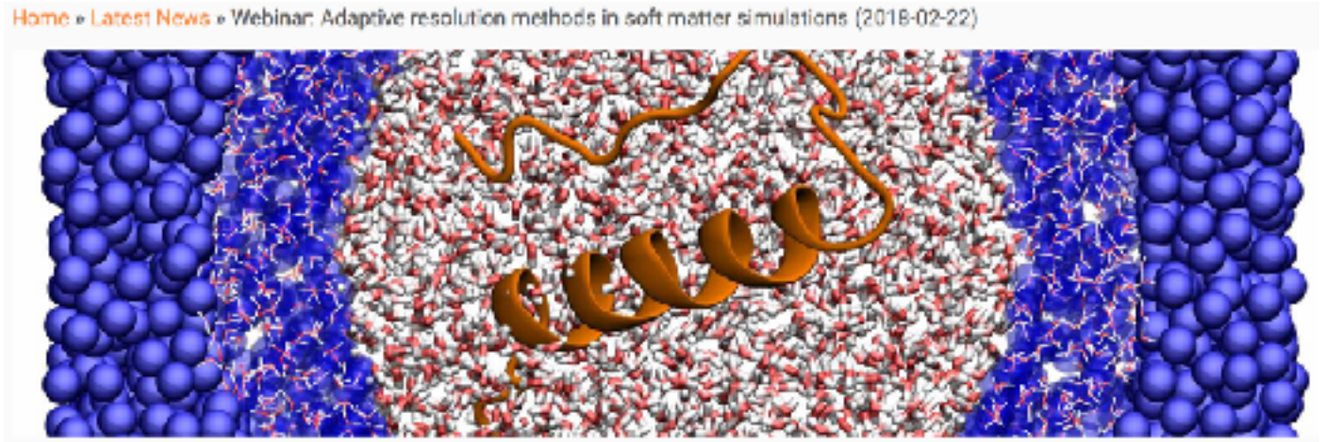
M. Eric Irrgang[1,2], Jennifer M. Hays[1,2] and Peter M. Kasson[1,2,*]

[1]Department of Biomedical Engineering and [2]Department of Molecular Physiology and Biological Physics, University of Virginia, Charlottesville, VA 22908, USA

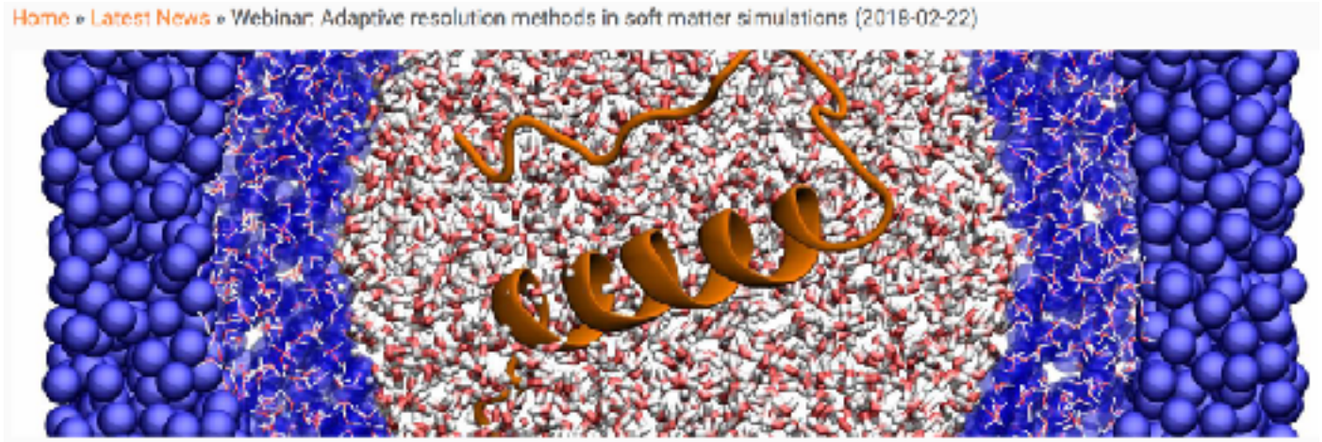
*To whom correspondence should be addressed.
Associate Editor: Alfonso Valencia

Received on March 15, 2018; revised on May 23, 2018; editorial decision on June 8, 2018; accepted on June 12, 2018

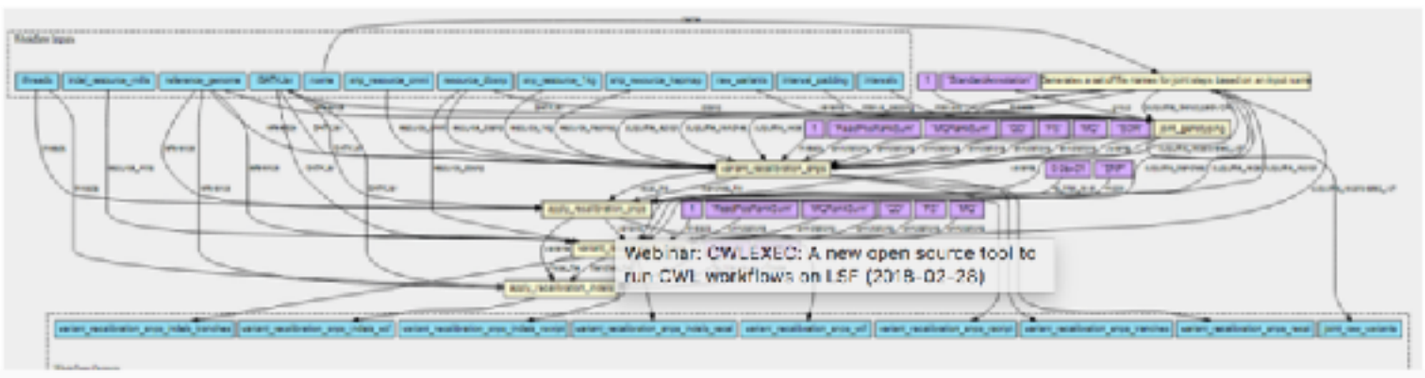
Need: API access to MD tools



Need: API access to MD tools

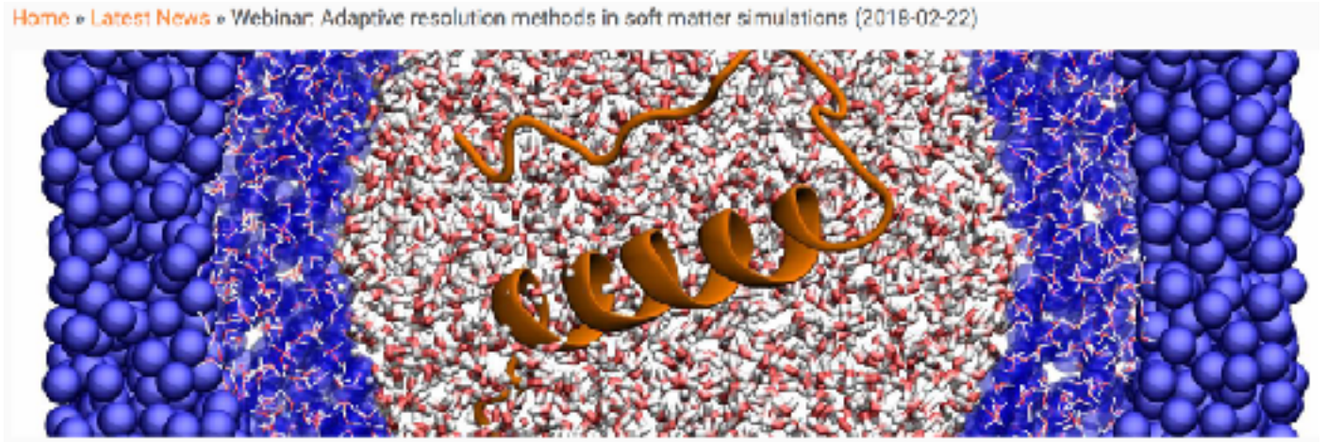


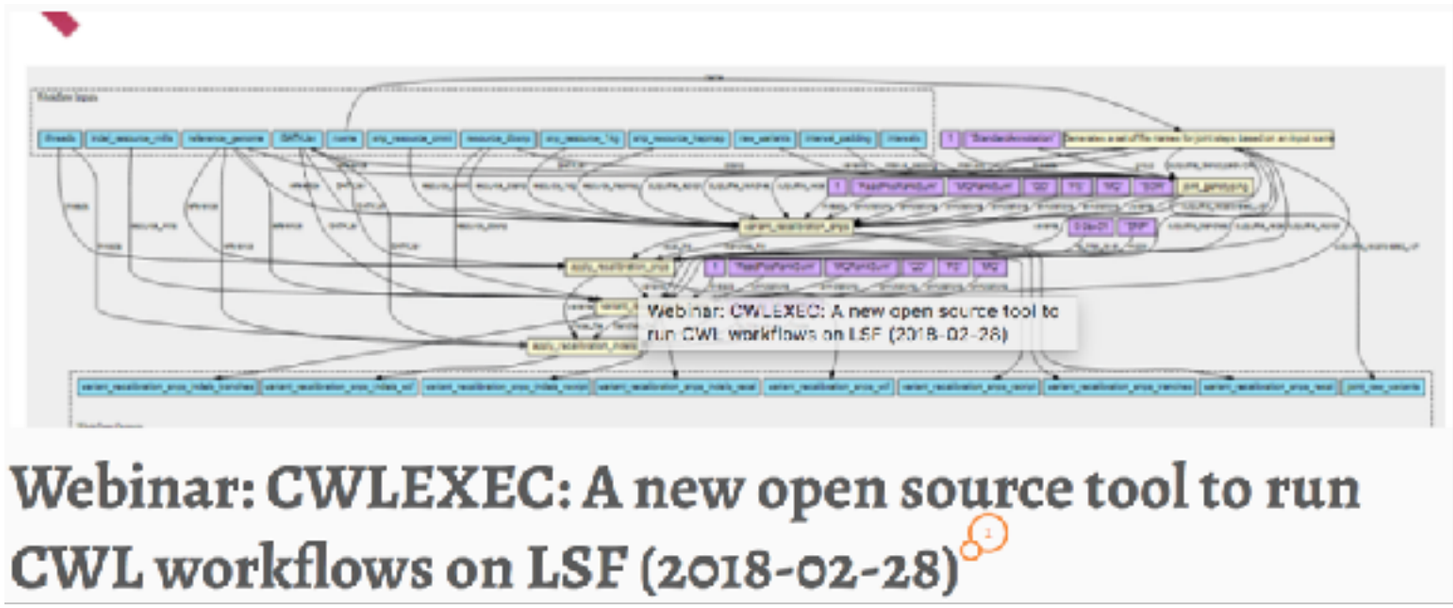
Webinar: Adaptive resolution methods in soft matter simulations (2018-02-22)

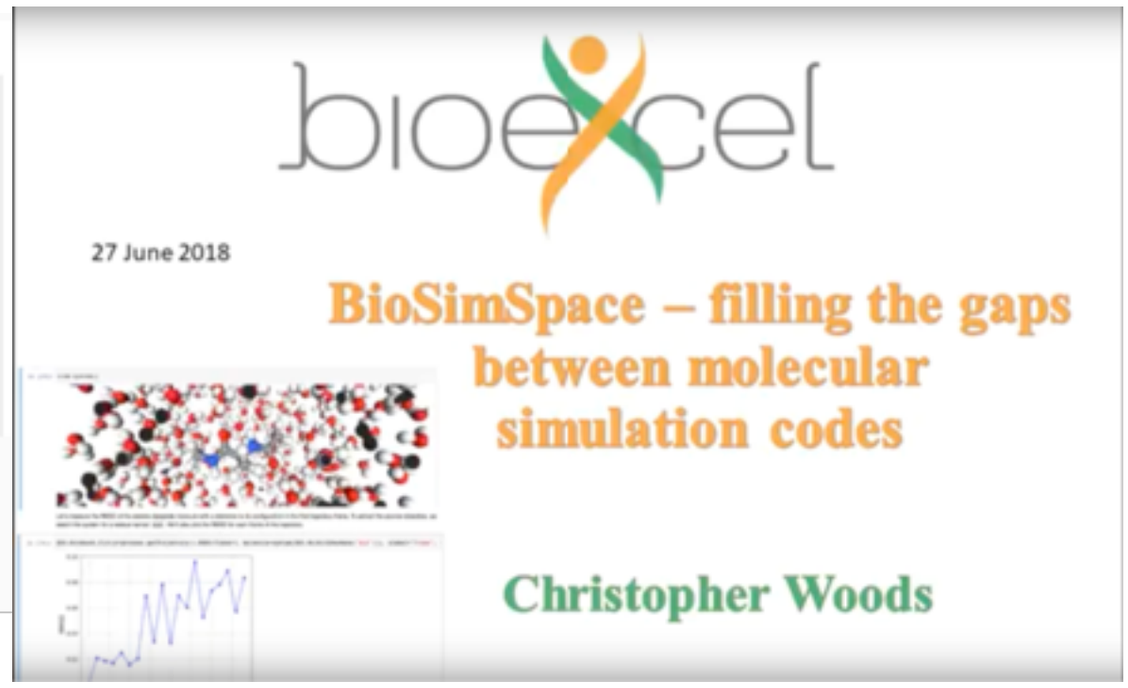


Webinar: CWLEXEC: A new open source tool to run CWL workflows on LSF (2018-02-28)

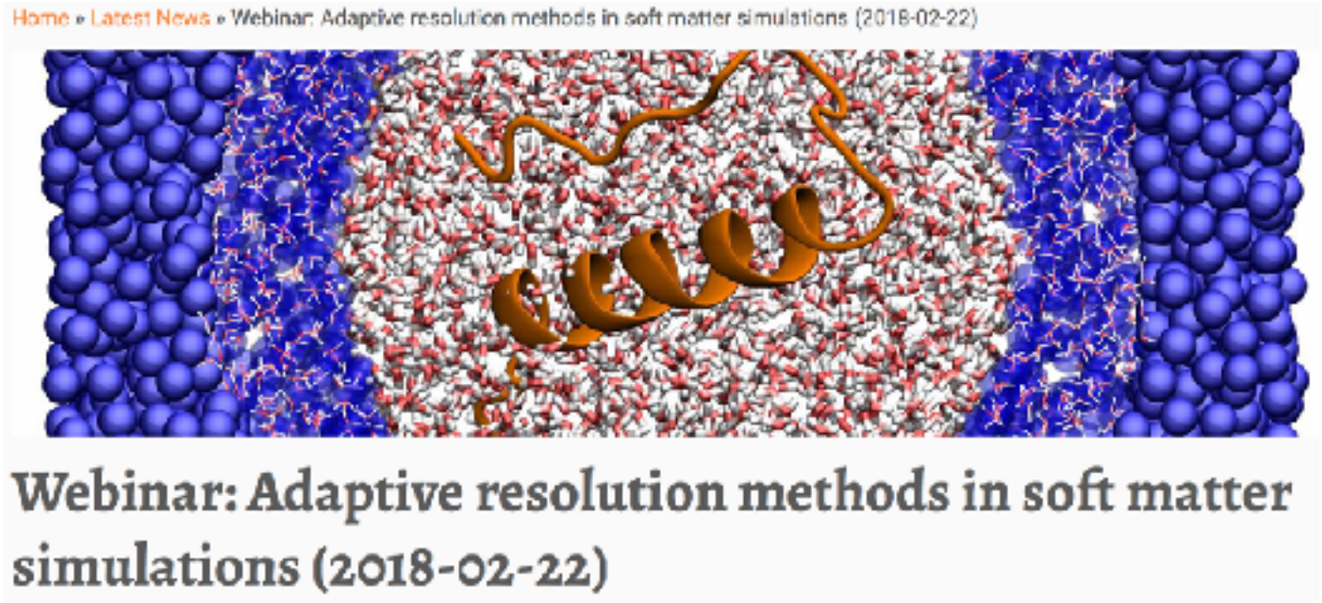
Need: API access to MD tools

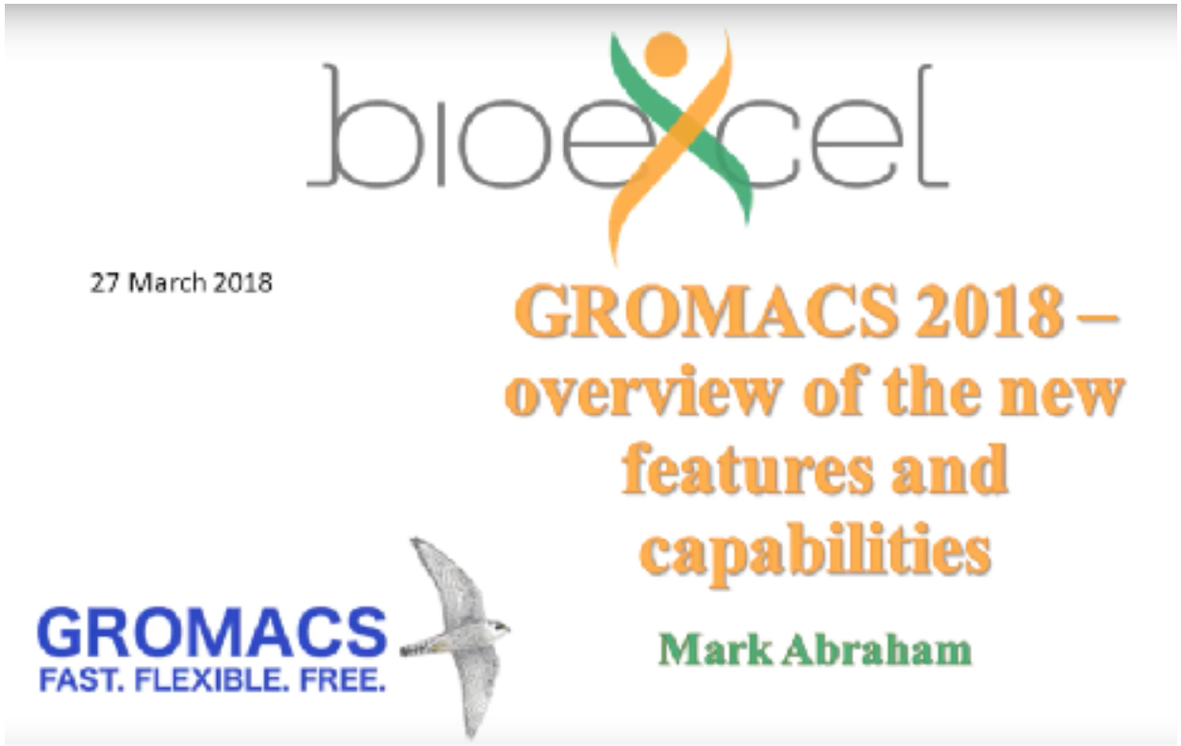


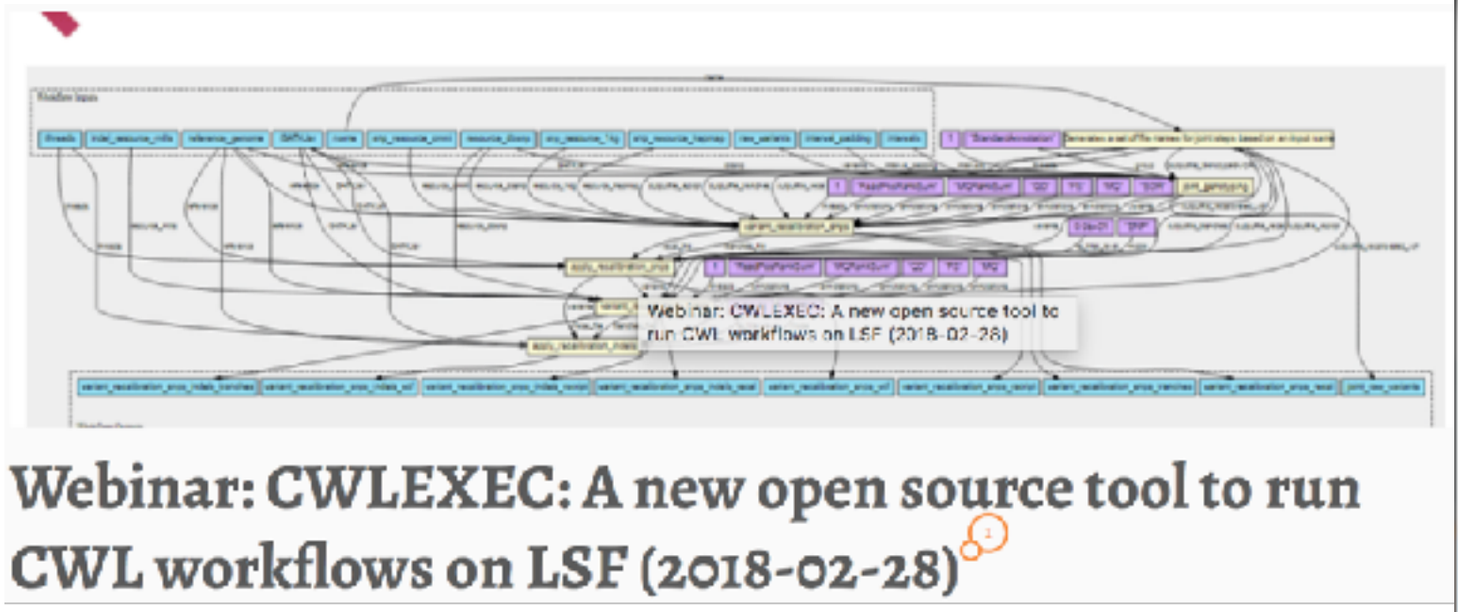


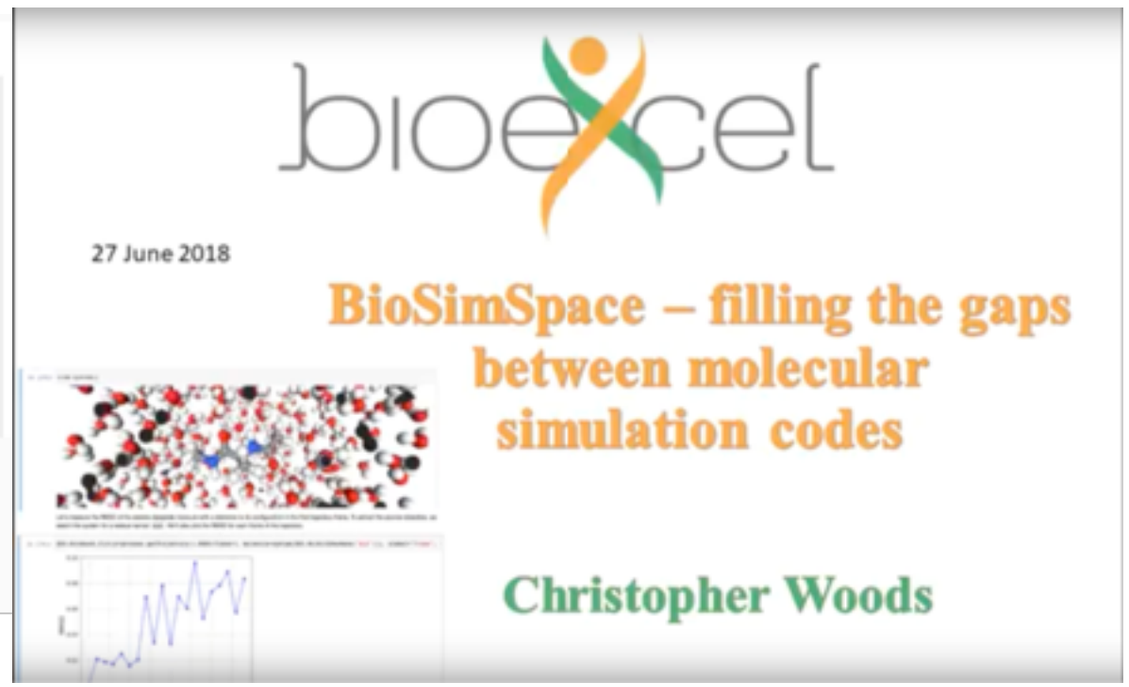


Need: API access to MD tools

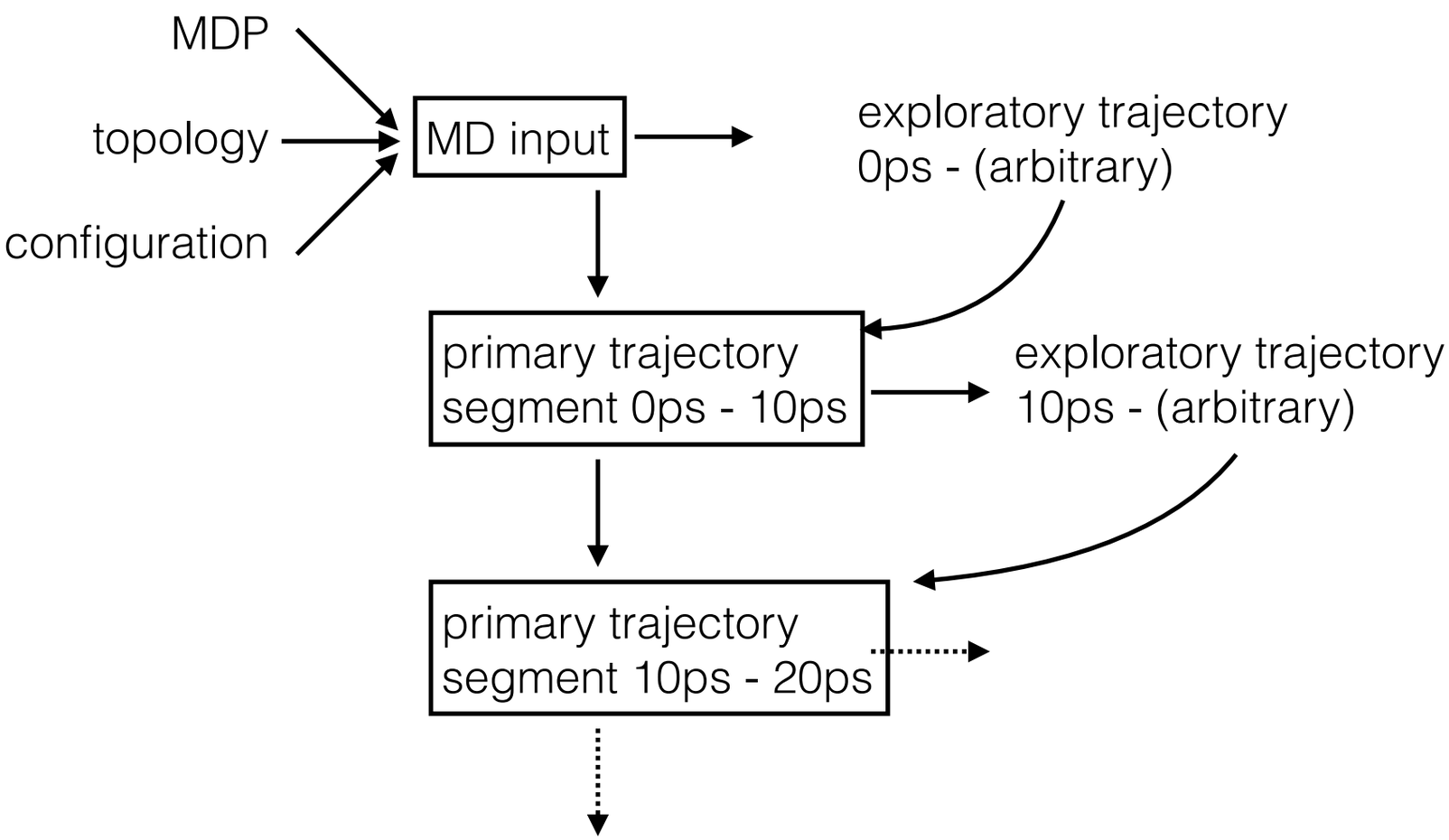




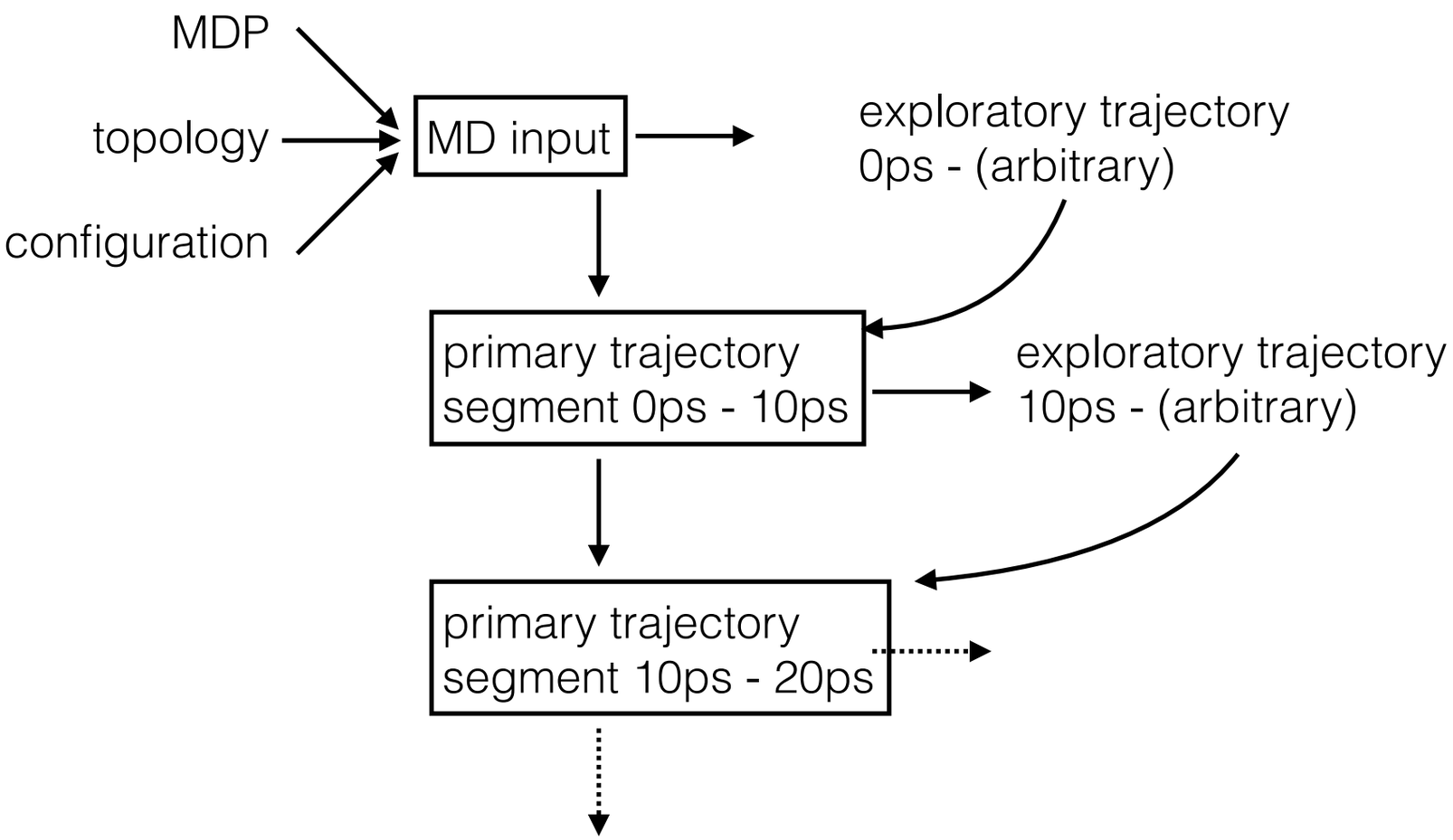


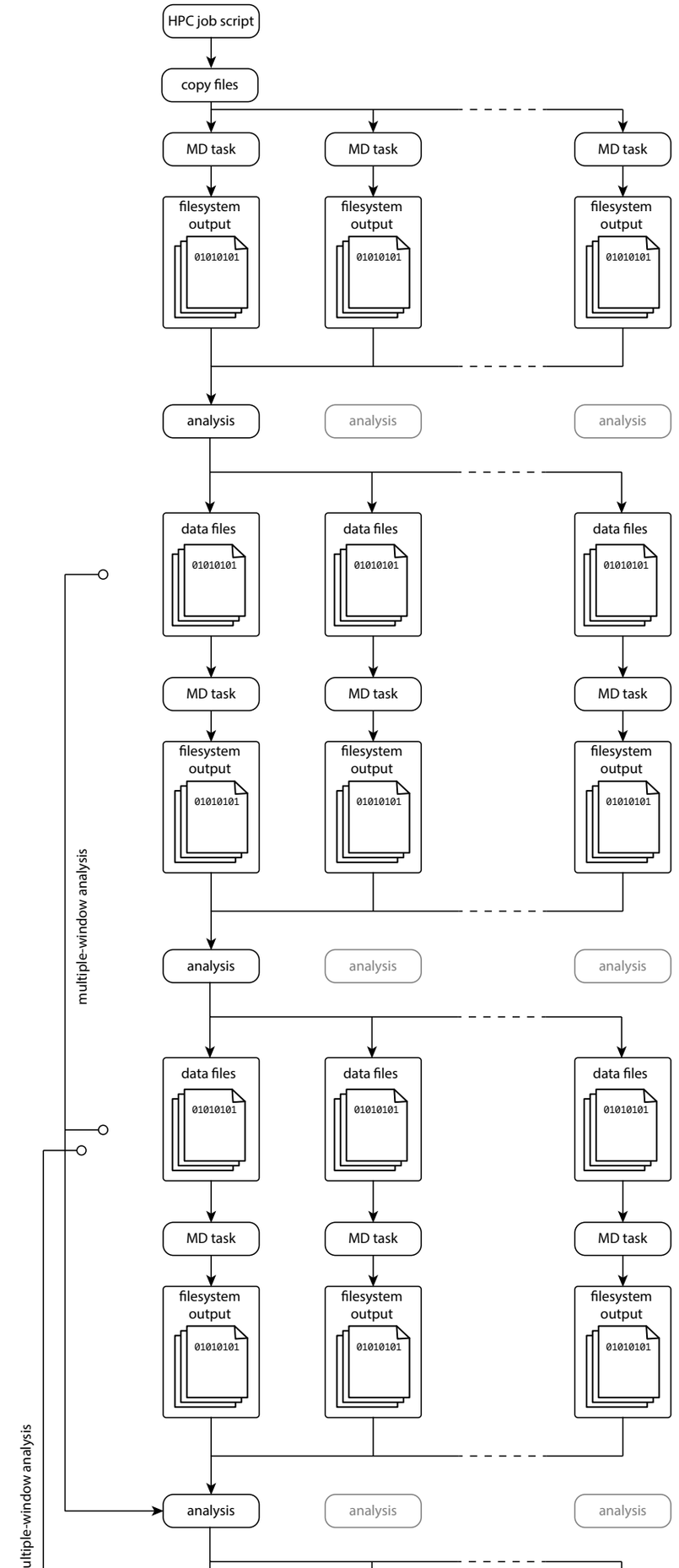


Sample scenarios



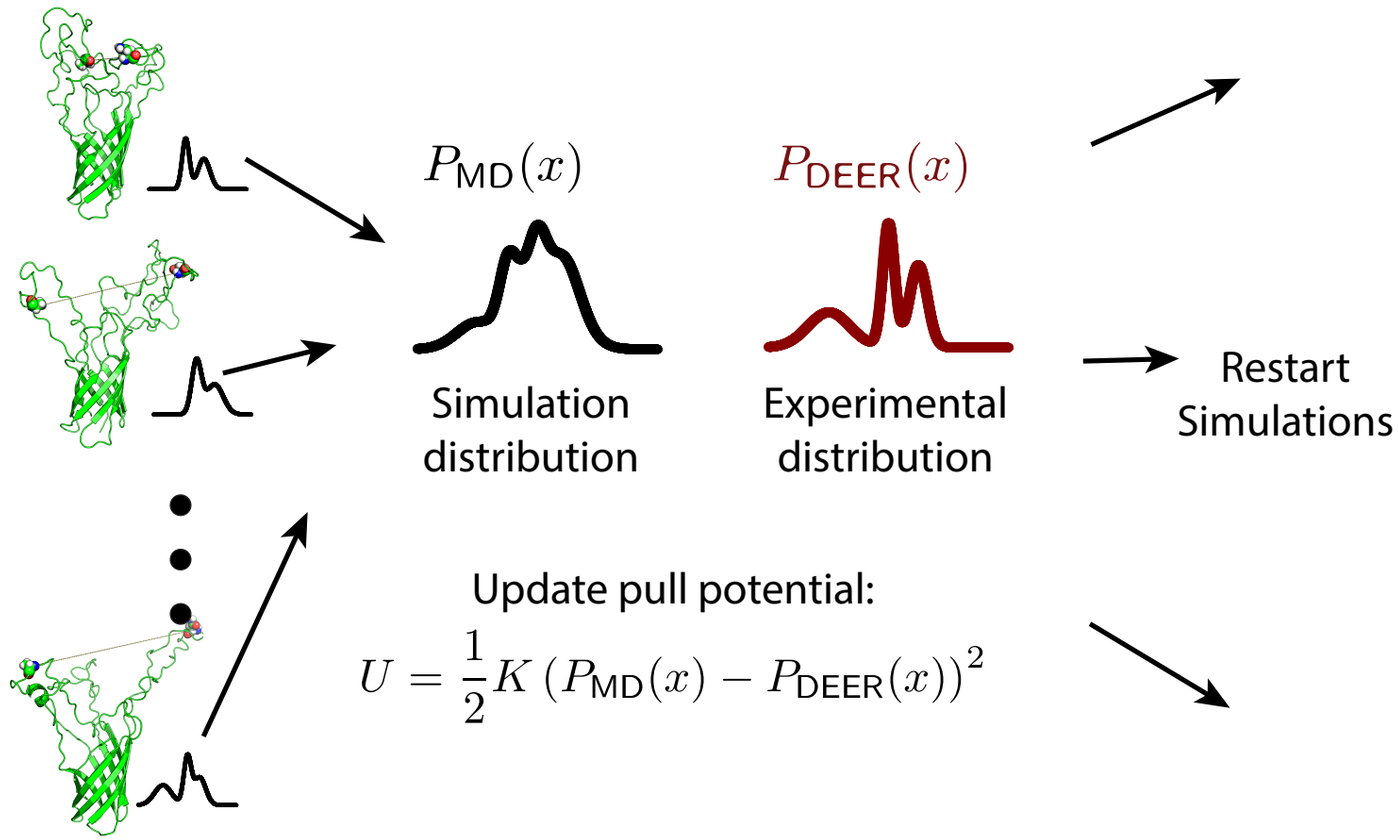
Sample scenarios





Complex workflows and custom code

restrained ensemble

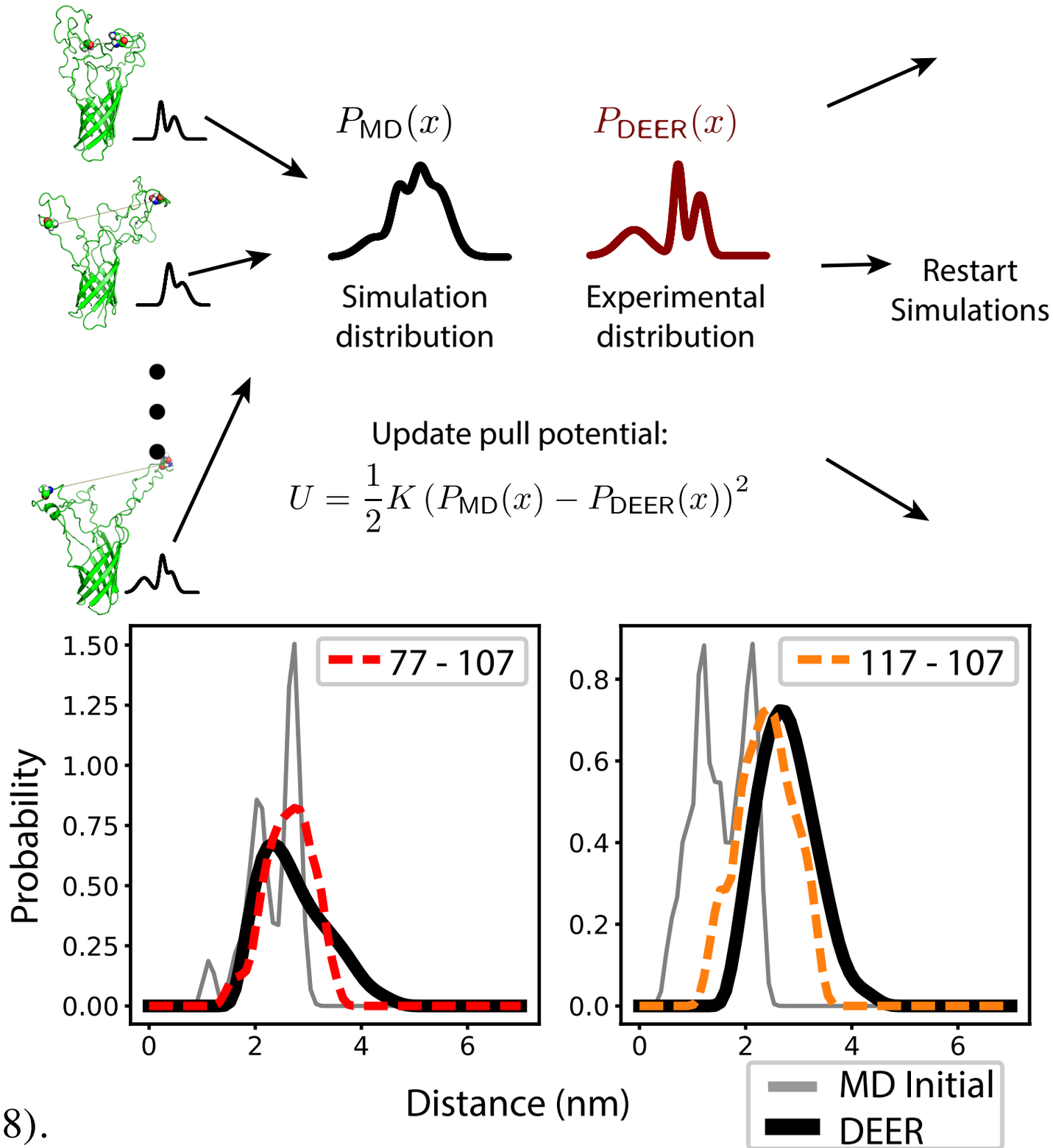


Irrgang, Hays, & Kasson. *Bioinformatics* (2018).

DOI: 10.1093/bioinformatics/bty484

Complex workflows and custom code

restrained ensemble



Irrgang, Hays, & Kasson. *Bioinformatics* (2018).

DOI: 10.1093/bioinformatics/bty484

Building the simulation

a

```
>>> md = gmx.from_file([filename1, filename2, filename3, ...])
```

b

```
>>> potential = myplugin.EnsembleRestraint(sites, *args, **kwargs)
```

```
>>> md.add_dependancy(potential)
```

c

```
>>> gmx.run()
```

Irrgang, Hays, & Kasson. *Bioinformatics* (2018).

DOI: 10.1093/bioinformatics/bty484

Specifying work

Python command → **Work graph**

a

```
>>> md = gmx.from_file([filename1, filename2, filename3, ...])
```

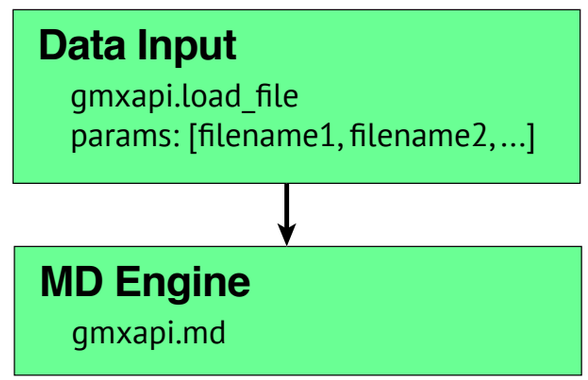
b

```
>>> potential = myplugin.EnsembleRestraint(sites, *args, **kwargs)
```

```
>>> md.add_dependancy(potential)
```

c

```
>>> gmx.run()
```



Irrgang, Hays, & Kasson. *Bioinformatics* (2018).

DOI: 10.1093/bioinformatics/bty484

Specifying work

Python command → Work graph

a

```
>>> md = gmx.from_file([filename1, filename2, filename3, ...])
```

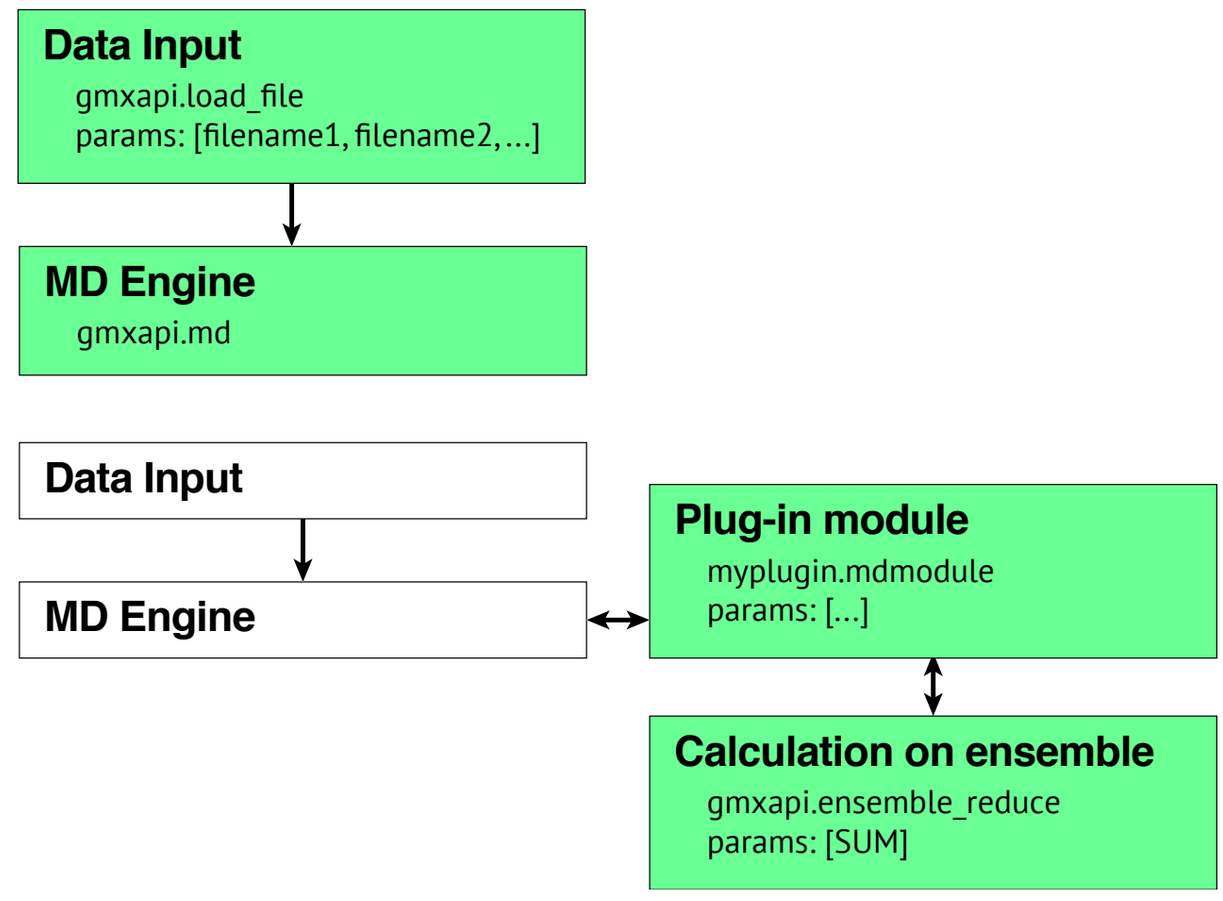
b

```
>>> potential = myplugin.EnsembleRestraint(sites, *args, **kwargs)
```

```
>>> md.add_dependancy(potential)
```

c

```
>>> gmx.run()
```



Irrgang, Hays, & Kasson. *Bioinformatics* (2018).

DOI: 10.1093/bioinformatics/bty484

Dispatching for execution

Python command → **Work graph**

a

```
>>> md = gmx.from_file([filename1, filename2, filename3, ...])
```

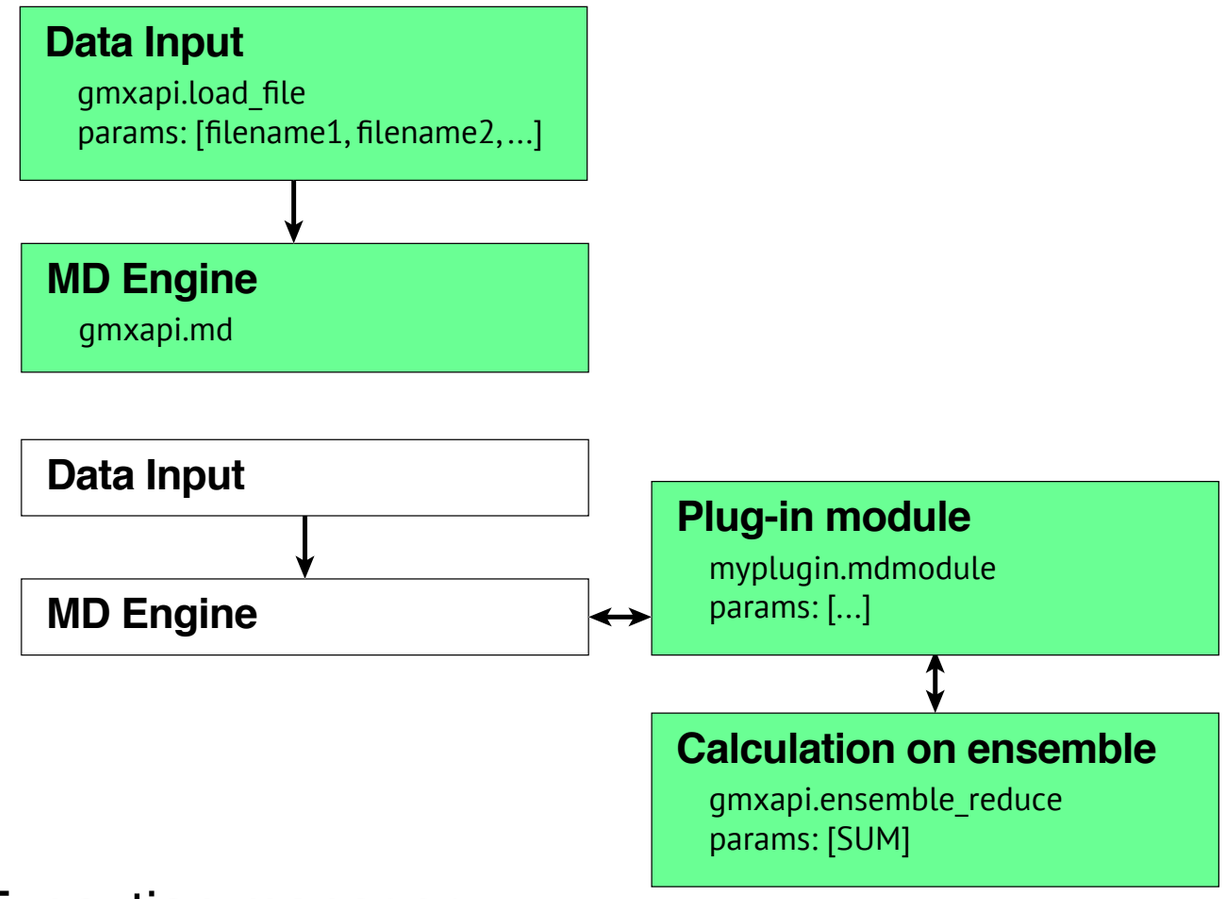
b

```
>>> potential = myplugin.EnsembleRestraint(sites, *args, **kwargs)
```

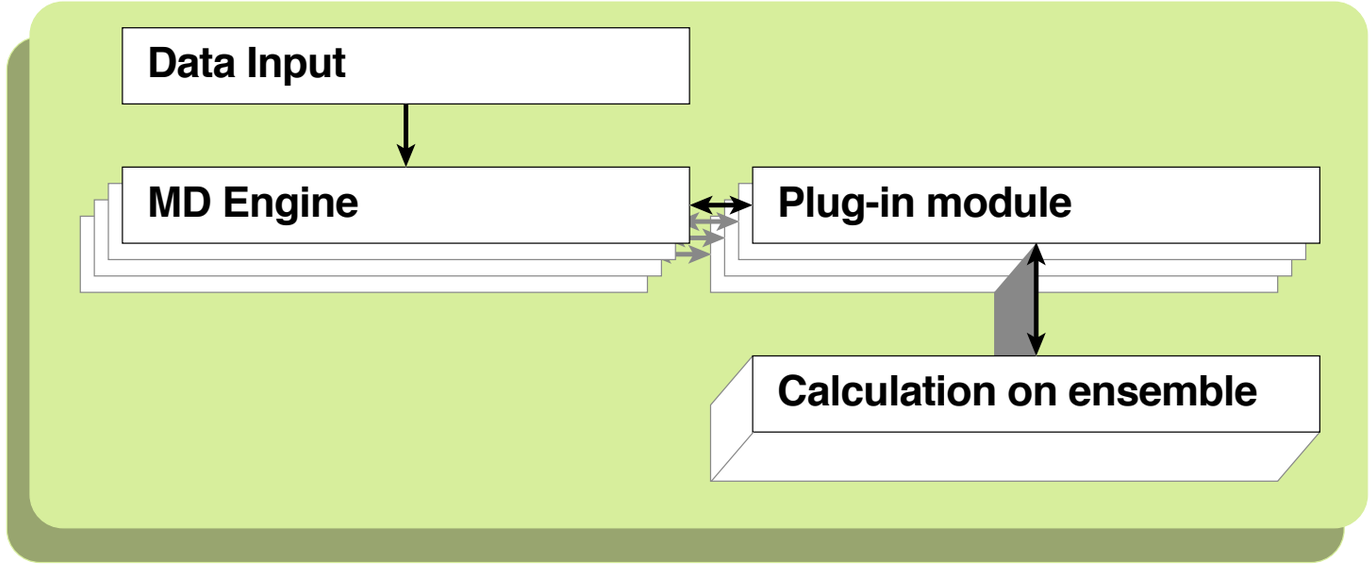
```
>>> md.add_dependancy(potential)
```

c

```
>>> gmx.run()
```



Execution manager



Irrgang, Hays, & Kasson. *Bioinformatics* (2018).

DOI: 10.1093/bioinformatics/bty484

Work specification schema

Python command

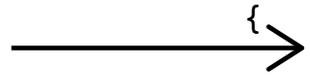
a

```
>>> md = gmx.from_file([filename1, filename2, filename3, ...])
```

b

```
>>> potential = myplugin.EnsembleRestraint(sites, *args, **kwargs)
```

```
>>> md.add_dependancy(potential)
```



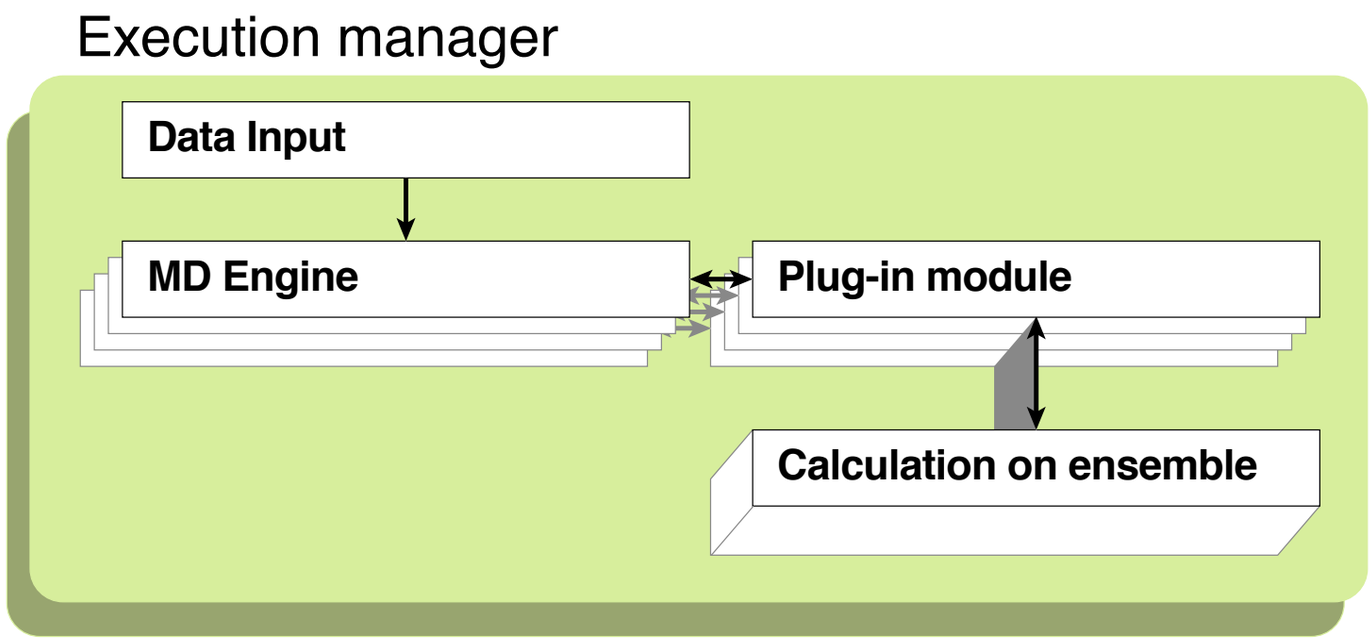
```
{
"version": "gmxapi_workspec_0_1",
"elements":
{
    “tpr_input”:
    {
        “namespace”: “gmxapi”,
        “operation”: “load_tpr”,
        “params”: […],
        “depends”: []
    }
    “md_sim”:
    {
        “namespace”: “gmxapi”,
        “operation”: “md”,
        “params”: [],
        “depends”: [“tpr_input”, “ensemble_restraint”]
    }
    “ensemble_restraint_1”:
    {
        “namespace”: “myplugin”,
        “operation”: “ensemble_restraint”,
        “params”: […],
        “depends”: []
    }
}
}
```

Irrgang, Hays, & Kasson. *Bioinformatics* (2018).

DOI: 10.1093/bioinformatics/bty484

Middleware layer

```
{
    "version": "gmxapi_workspec_0_1",
    "elements":
    {
        “tpr_input”:
        {
            “namespace”: “gmxapi”,
            “operation”: “load_tpr”,
            “params”: […],
            “depends”: []
        }
        “md_sim”:
        {
            “namespace”: “gmxapi”,
            “operation”: “md”,
            “params”: [],
            “depends”: [“tpr_input”, “ensemble_restraint”]
        }
        “ensemble_restraint_1”:
        {
            “namespace”: “myplugin”,
            “operation”: “ensemble_restraint”,
            “params”: […],
            “depends”: []
        }
    }
}
```



Irrgang, Hays, & Kasson. *Bioinformatics* (2018).

DOI: 10.1093/bioinformatics/bty484

Restrained Ensemble Simulation

a

```
>>> md = gmx.from_file([filename1, filename2, filename3, ...])
```

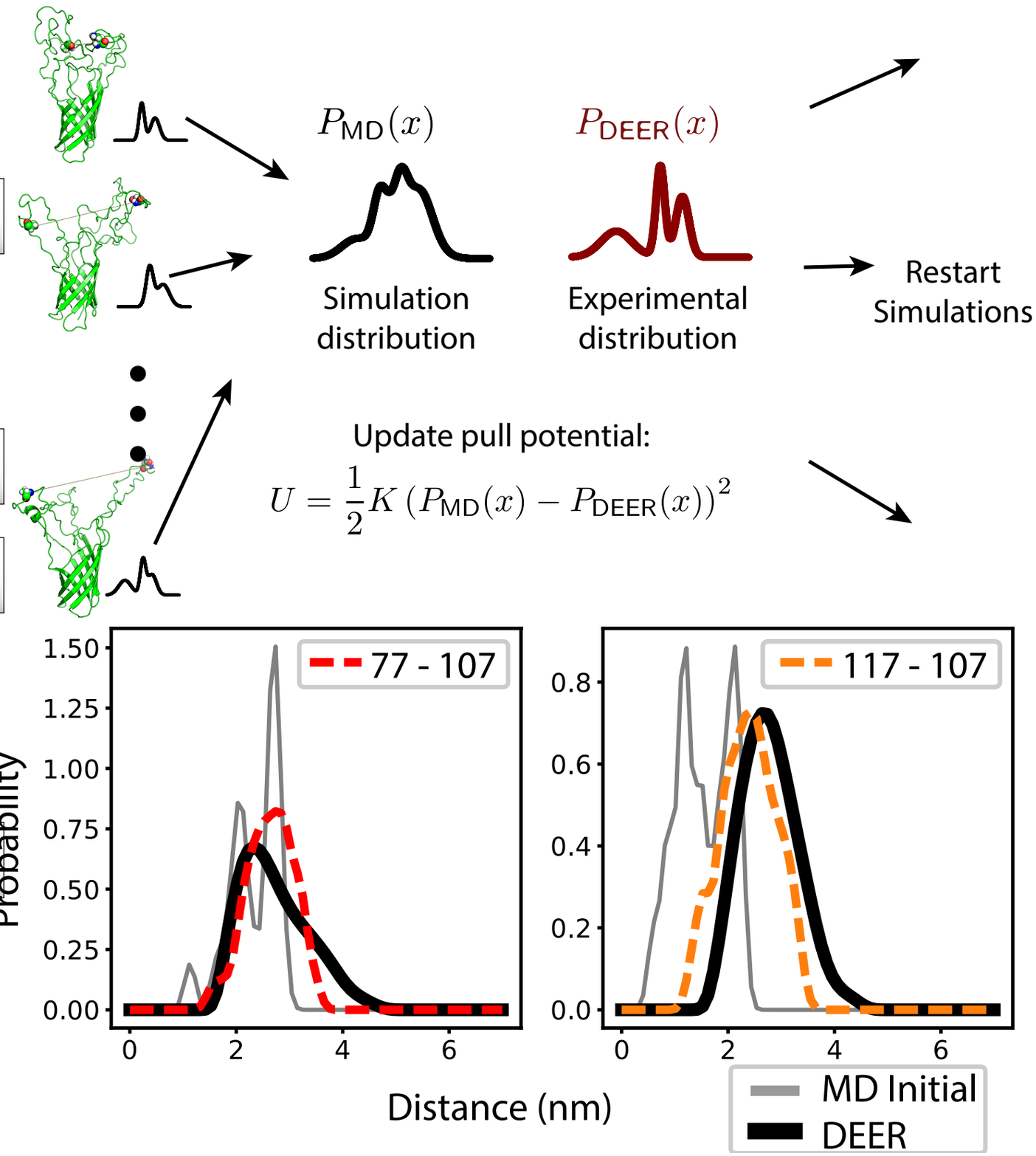
b

```
>>> potential = myplugin.EnsembleRestraint(sites, *args, **kwargs)
```

```
>>> md.add_dependancy(potential)
```

c

```
>>> gmx.run()
```



Irrgang, Hays, & Kasson. *Bioinformatics* (2018).

DOI: 10.1093/bioinformatics/bty484

Examples

Once the `gmxapi` package is installed, running simulations is easy with `gmx.workflow.from_tpr()` and `gmx.run()`.:

```
import gmx
md = gmx.workflow.from_tpr(tpr_filename)
gmx.run(md)
```

To run a batch of simulations, just pass an array of inputs.:

```
import gmx
md = gmx.workflow.from_tpr([tpr_filename1, tpr_filename2, ...])
gmx.run(md)
```

If additional arguments need to be provided to the simulation as they would for the `mdrun` command line tool, you can add them to the workflow specification when you create the MD work element.:

```
md = gmx.workflow.from_tpr(tpr_list,
                           tmpi=20,
                           grid=[3, 3, 2],
                           pme_threads_per_rank=1,
                           pme_ranks=2,
                           threads_per_rank=1)
```

https://github.com/kassonlab/gmxapi

Full script

If you have written plugins or if you have downloaded and built the sample plugin, you attach it to your workflow by making it a dependency of the MD element. You can use the `add_dependency()` member function of the `gmx.workflow.WorkElement` returned by `from_tpr()`. The following example applies a harmonic spring restraint between atoms 1 and 4:

```
import gmx
import myplugin
assert gmx.version.is_at_least(0,0,6)

md = gmx.workflow.from_tpr([tpr_filename])
params = {'sites': [1, 4],
          'R0': 2.0,
          'k': 10000.0}
potential_element = gmx.workflow.WorkElement(namespace="myplugin",
                                             operation="create_restraint",
                                             params=params)
potential_element.name = "harmonic_restraint"
md.add_dependency(potential_element)
gmx.run(md)
```

https://github.com/kassonlab/gmxapi

gmxapi 0.0.6

https://github.com/kassonlab/gmxapi

Change Log¶

0.0.6

Interface and feature updates

- Updates to `gmx.version` module
- Automatically set and restore from MD simulation checkpoints in the session working directory.
- Allow control of whether simulation output is appended or truncated (PR #126).
- Allow plugins to issue a stop signal to MD simulations (reference #62 for gromacs-gmxapi and sample_restraint repos).
- Changes to `gmx.exceptions`
- Allow full CMake-driven install
- Updated example notebooks in sample_restraint repository.

Internal

- Improved CI testing
- #64 Unique work spec identification.

Bug fixes

- #66 Docker does not access current gmxpy version.
- #123 Race condition in session closing.

Better data flow (future)

```
>>> my_stop_condition = gmx.logical_and(potential1.stop, potential2.stop)
>>> md = gmx.workflow.from_tpr([tpr_filename, tpr_filename],
                               restraint=[potential1, potential2],
                               stop=[my_stop_condition],
                               override_nsteps=True)
>>> potential3 = myplugin.new_restraint(alpha=potential1.output.alpha)
>>> # or
>>> # potential3 = myplugin.new_restraint(params=potential1.output.params)
```

The API hook can be provided to the C++ plugin as a function pointer in the Resources object. At the higher level, the MD element params will look like

```
`params`: {
    `input`:{
        `restraint`: ["potential1.interface.restraint", "potential2.interface.restraint"],
        `stop`: ["my_stop_condition.ostream"]
    }
}
```

where "my_stop_condition" is an element for the `gmxapi.logical_and` operation, with `params` equal to

```
`input`: ["potential1.ostream.stop", "potential2.ostream.stop"]
```

Architecture and protocols

```
system.def(
    "add_mdmodule",
    [](System* system, py::object force_object){
            auto spec = system->getSpec();
            auto holder = new gmxapi::MDHolder(spec);
            holder->name_ = "pygmx holder";
            auto deleter = [](PyObject *o) {
                if (PyCapsule_IsValid(o, gmxapi::MDHolder_Name))
                {
                    auto holder_ptr = (gmxapi::MDHolder *) PyCapsule_GetPointer(o, gmxapi::MDHolder_Name);
                    delete holder_ptr;
               };
            };
            auto capsule = py::capsule(holder,
                                       gmxapi::MDHolder_Name,
                                       deleter);
            py::object bind = force_object.attr("bind");
            py::object obj = capsule;
            bind(obj);
   };
```

```
gmxapi_mdmodule.def(
    "bind",
    [](std::shared_ptr<TestModule> self, py::object object){
            auto holder = (gmxapi::MDHolder*) PyCapsule_GetPointer(
                object.ptr(),
                gmxapi::MDHolder::api_name);
            auto spec = holder->getSpec();
            spec->addModule(self);
   }
);
```

```
gmx::PotentialPointData EnsembleHarmonic::calculate(gmx::Vector v,
                                                    gmx::Vector v0,
                                                    double t);
```

gmxapi: a high-level interface for advanced control and extension of molecular dynamics simulations.

Irrgang, M. Eric, Hays, Jennifer M., & Kasson, Peter M. (2018).
Bioinformatics. https://doi.org/10.1093/bioinformatics/bty484

National Institutes of Health R01GM115790 to P.M.K,
in collaboration with

- Pascal Merz & Michael Shirts, U.C. Boulder
- Mark Abraham, KTH Stockholm

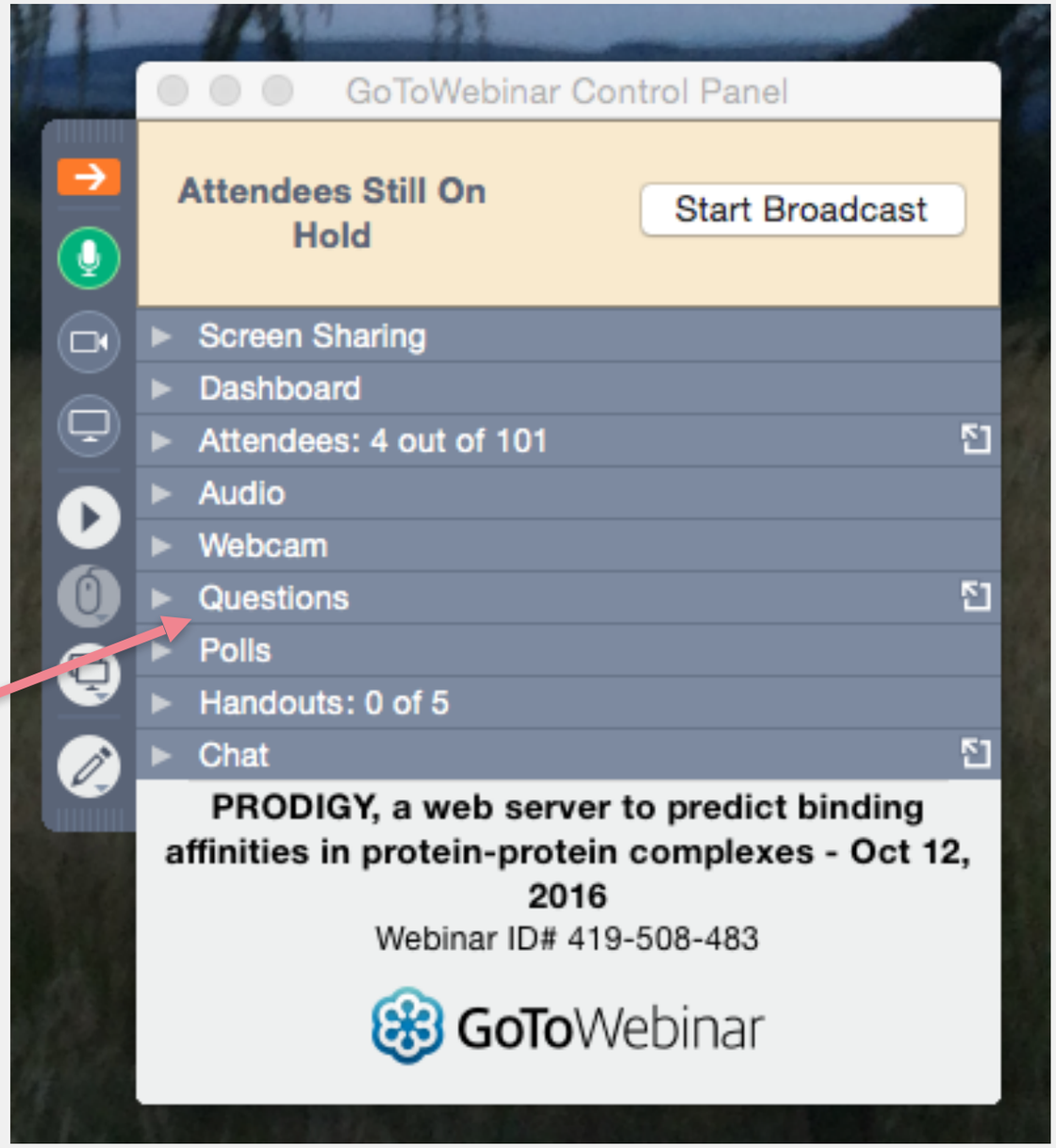
MolSSI fellowship to M.E.I through subaward to
National Science Foundation ACI1547580

https://github.com/kassonlab/gmxapi

Audience Q&A session

Please use the **Questions** function in GoToWebinar application

Any other questions or points to discuss after the live webinar? Join the discussion at http://ask.bioexcel.eu.



Status of "context" abstraction

`gmx.run()` provides the "big green go button" that users expect, but the hidden layers of abstraction are also accessible to users. gmxapi 0.1 specificies that gmx.run() will use or configure an appropriate execution context for the specified work, launch a session, and run until data flow for results is resolved. It is assumed to be essentially an alias for the following.

```
with gmx.get_context(simulation.workspec) as session:
    session.run()
```

The Context abstraction exists to allow modular handling of computing resources and environments. The user could specify non-default Context implementations or load Contexts from other Python modules.

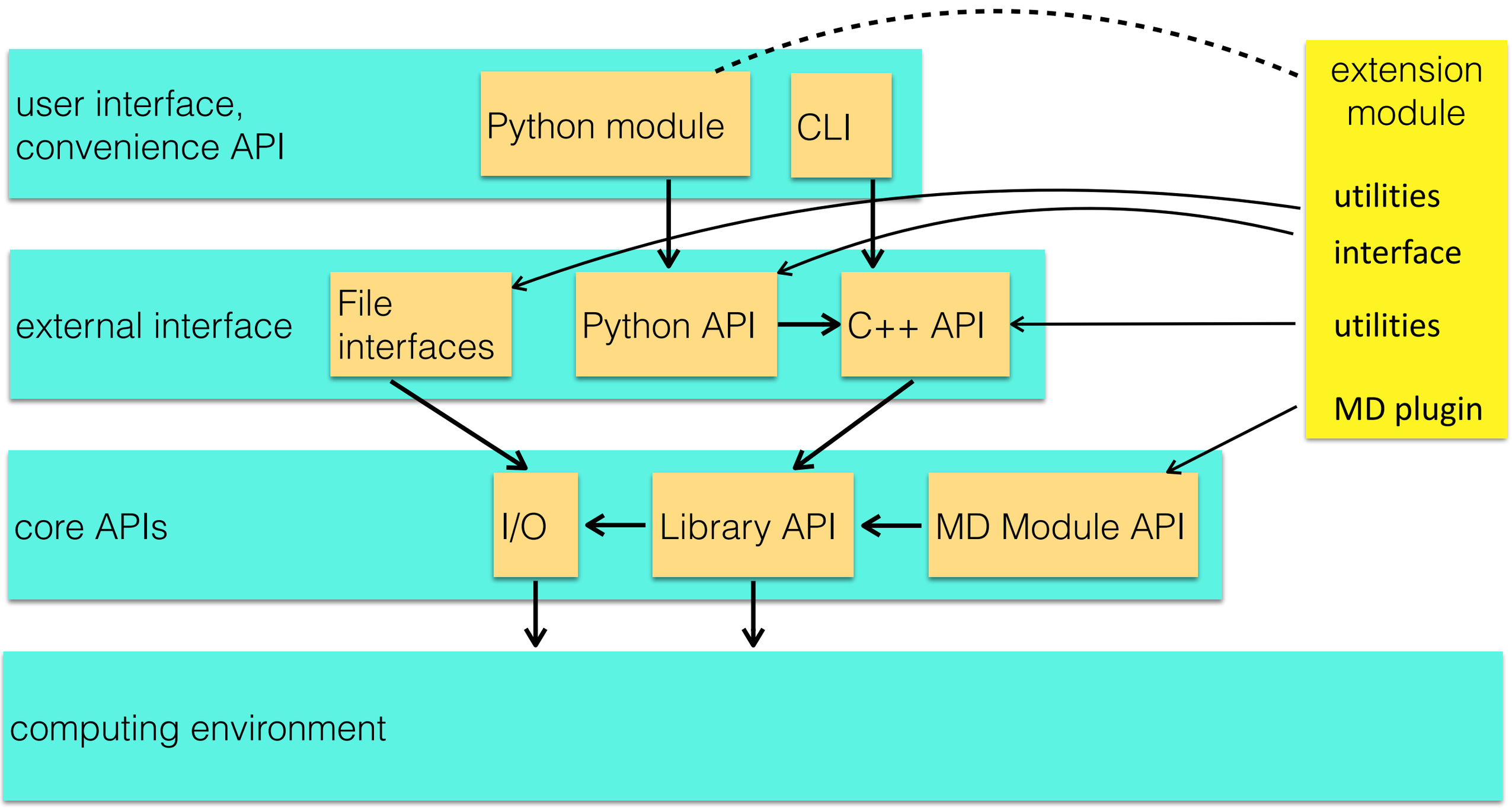
```
context = gmx.context.ParallelArrayContext(simulation)
with context as session:
    session.run()
```

Near term plans for Context extensions include

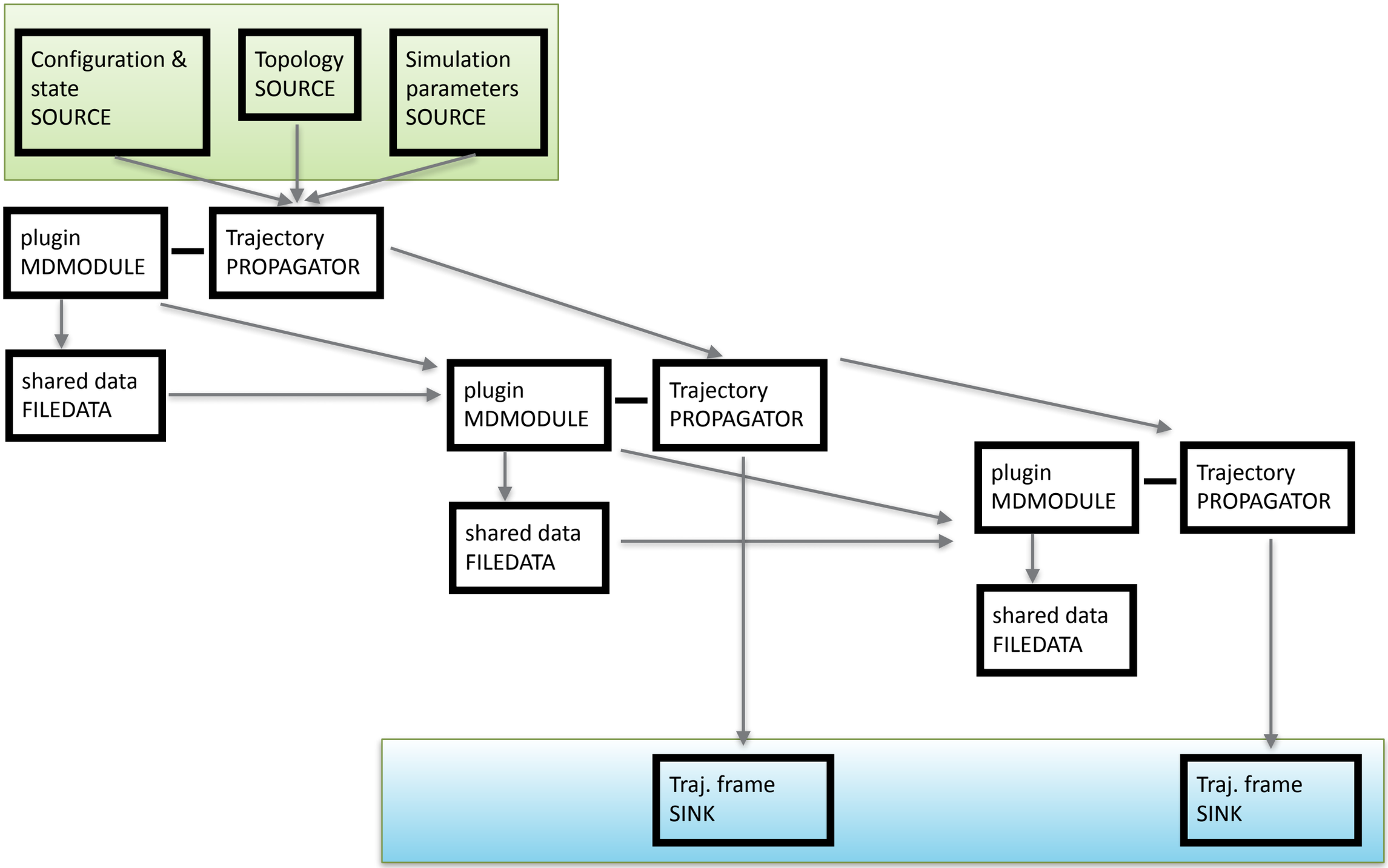
- support for tMPI or MPI GROMACS builds,
- serial execution fallback for workflows when MPI is not available
- remote SLURM execution
- container management

https://github.com/kassonlab/gmxapi

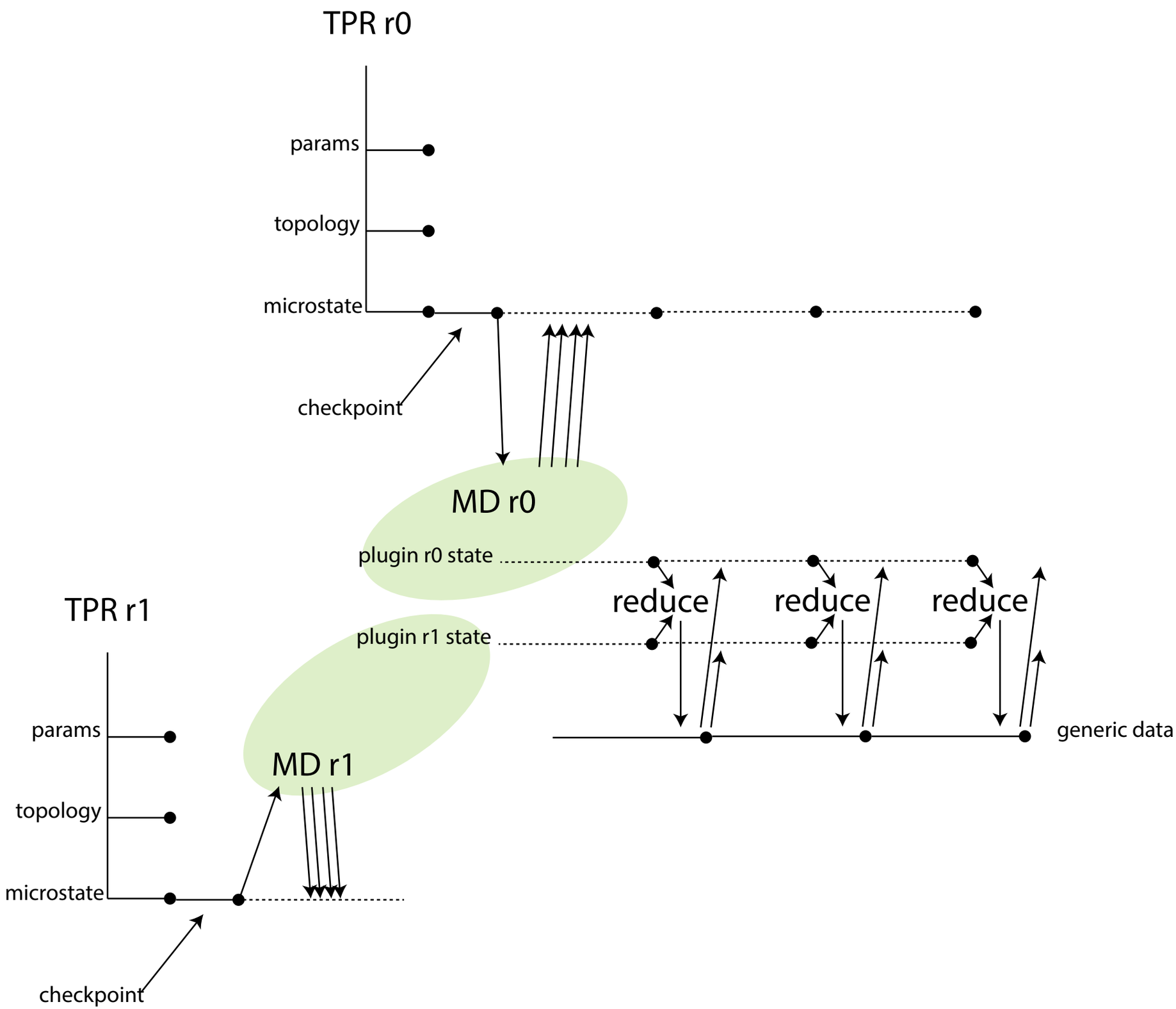
API compartmentalization



Directed, acyclic work graph



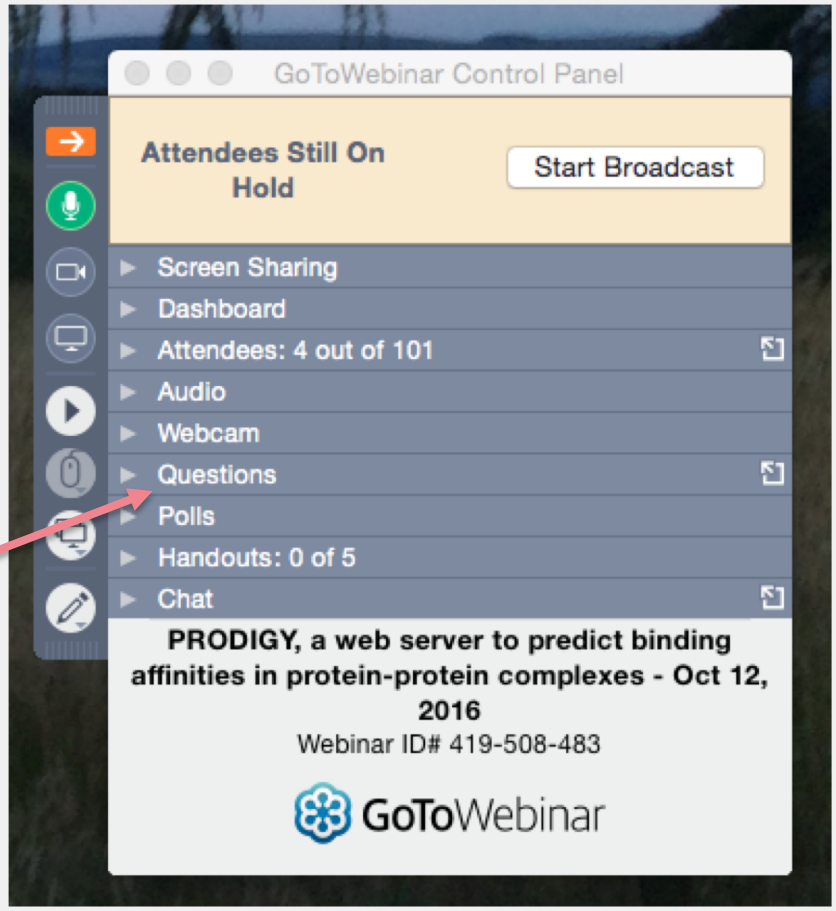
Data stream expression



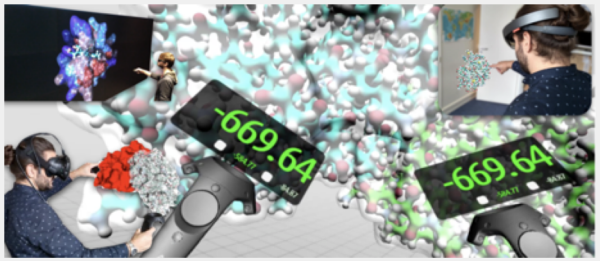
Audience Q&A session

Please use the **Questions** function in GoToWebinar application

Any other questions or points to discuss after the live webinar? Join the discussion the discussion at http://ask.bioexcel.eu.

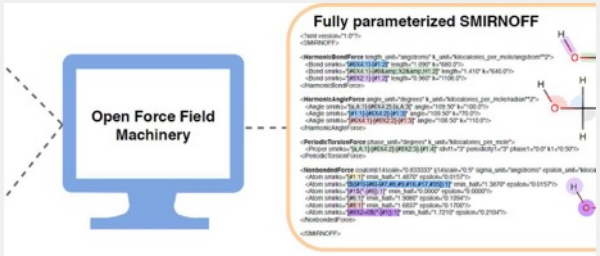


Coming up next



Immersive visual exploration of biomolecular systems in virtual reality – from static views to interactive dynamics **(2018-10-04)**

Presenter**: Marc Baaden**



Open Force Field Initiative: The SMIRNOFF format and learned chemical perception **(2018-10-10)**

Presenter**: Caitlin C. Bannan**