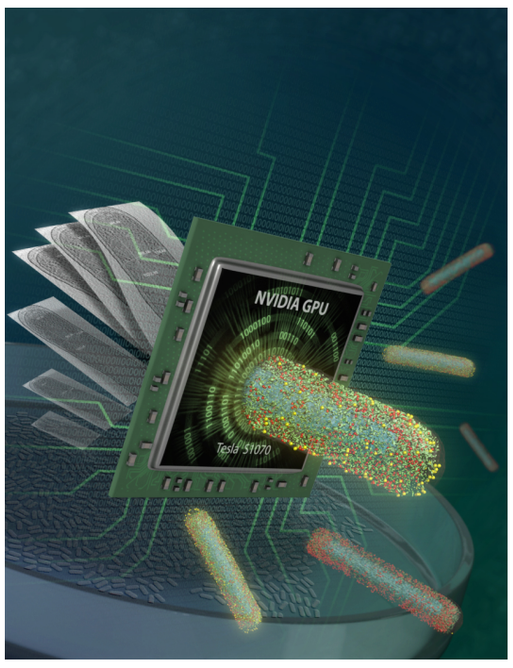
ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Part 2: Introduction to Using Lattice Microbes

Joseph R. Peterson and Michael J. Hallock
Luthey-Schulten Laboratory
University of Illinois at Urbana-Champaign

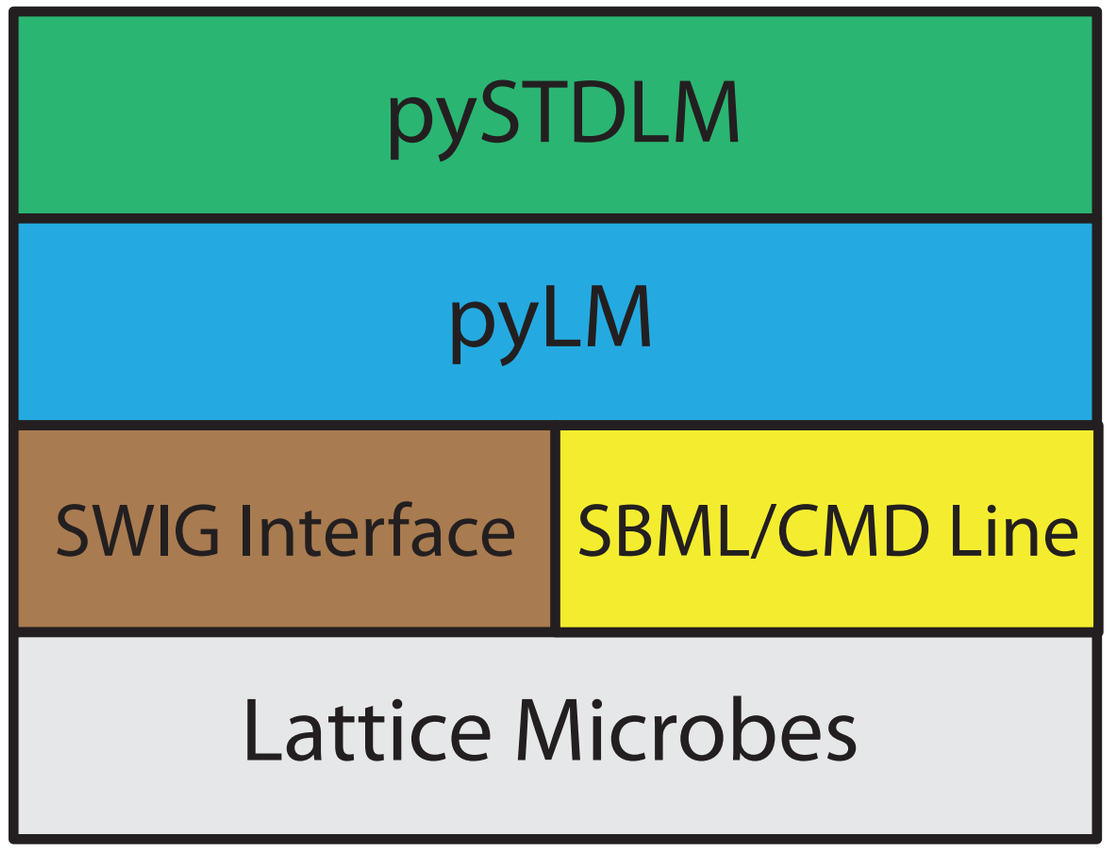


NIH Workshop on Computational Biophysics
November 17th, 2016
George Institute of Technology
Atlanta, GA, USA

NSF
Center for the Physics of Living Cells
NATIONAL INSTITUTES OF HEALTH
XSEDE
NCSA

Lattice Microbes

- Suite of well-stirred and spatial solvers
- GPU accelerated
- Model building and simulation analysis done in Python



Model Building with pyLM

1. Define chemical species

```
sim.defineSpecies(['minDadp', 'minDatp', 'minDm', 'minE', 'minDEm'])
```

2. Define spatial parameters

```
sim.buildCapsidCell(length=micron(4), diameter=micron(1), membraneThickness=nm(64))
```

3. Define reactions

```
membrane.addReaction( 'minDatp','minDm', rate=0.78)
membrane.addReaction(('minDatp','minDm'),('minDm','minDm'),rate=9e6*scalar)
```

4. Define diffusion kinetics

```
sim.setTransitionRate(species='minDatp', via='cytoplasm', to='membrane', rate=2.5e-12)
```

Model Building with pyLM

```
V  = 1.0e-15          # L
NA = 6.022e23         # molecules/mole
kf = 1.07e5/(NA*V) # /M/s
kr = 0.351            # /s
sim=CME.CMESimulation()
sim.defineSpecies(['A', 'B', 'C'])
sim.addReaction(reactant=('A','B'), product='C', rate=kf)
sim.addReaction(reactant='C', product=('A','B'), rate=kr)
sim.addParticles(species='A', count=1000)
sim.addParticles(species='B', count=1000)
sim.addParticles(species='C', count=0)
sim.setWriteInterval(microsecond(30))
sim.setSimulationTime(30)
sim.save(“bimolecular.lm”)
```

Running Simulations

Well-Stirred (CME)

```
lm -r 1-50 -ws -f cme.lm
```

Spatially resolved (RDME)

```
lm -r 1 -sp -f rdme.lm
```

Running Simulations

Well-Stirred (CME)

```
lm -r 1-50 -ws -f cme.lm
```

Number of replicate simulations to run

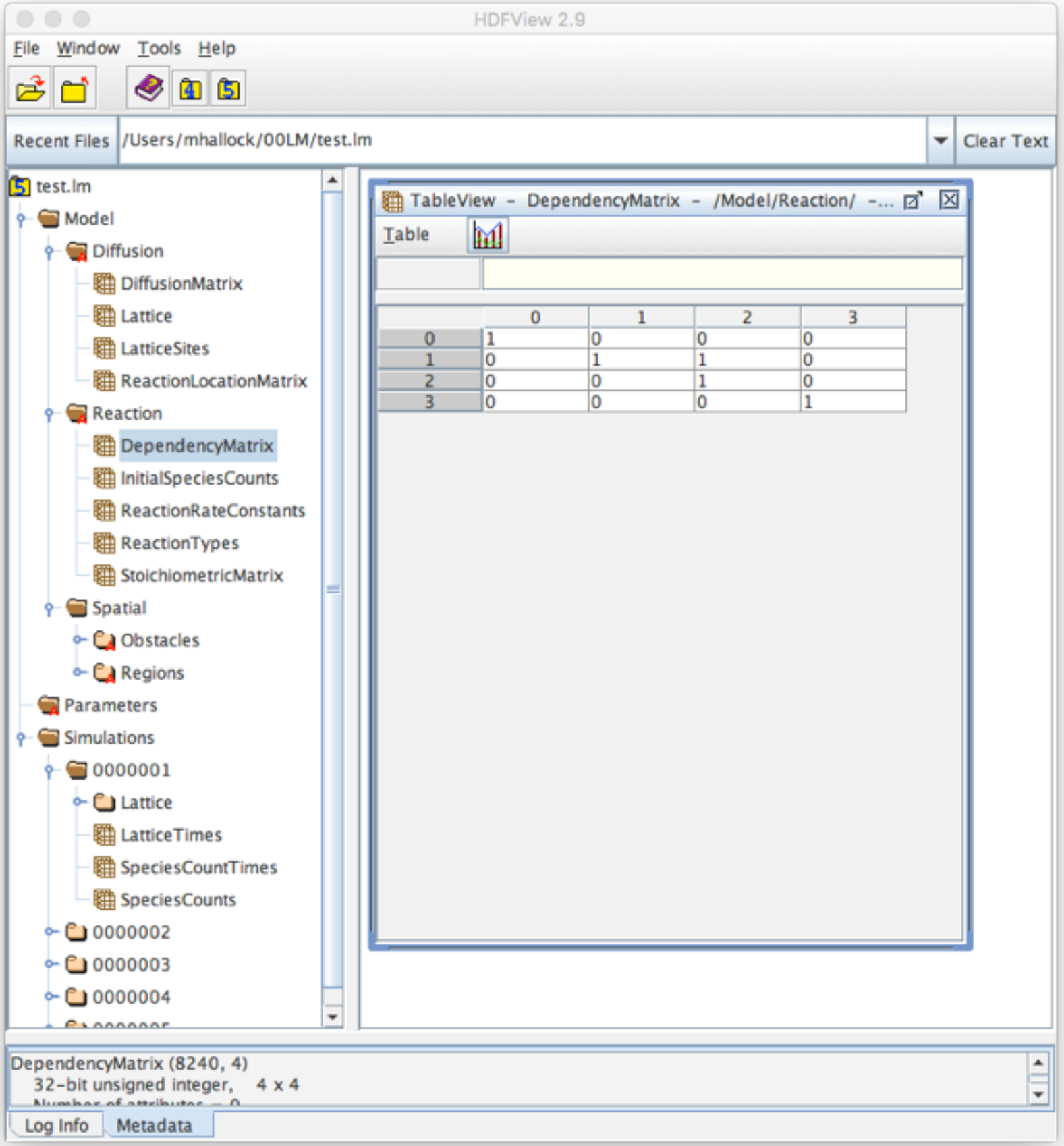
Spatially resolved (RDME)

```
lm -r 1 -sp -f rdme.lm
```

Input *and* Output file

LM Files

All model input data is stored alongside the simulation results in the same file

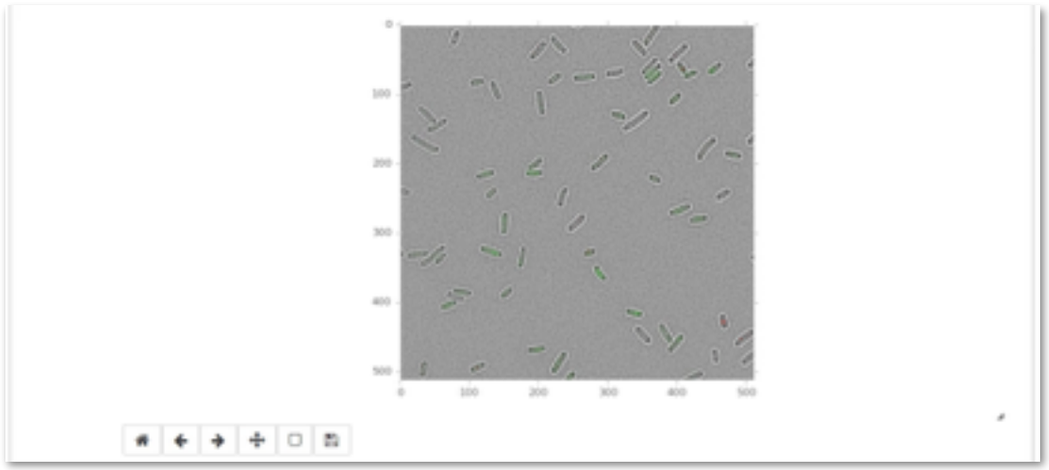


Installing LM

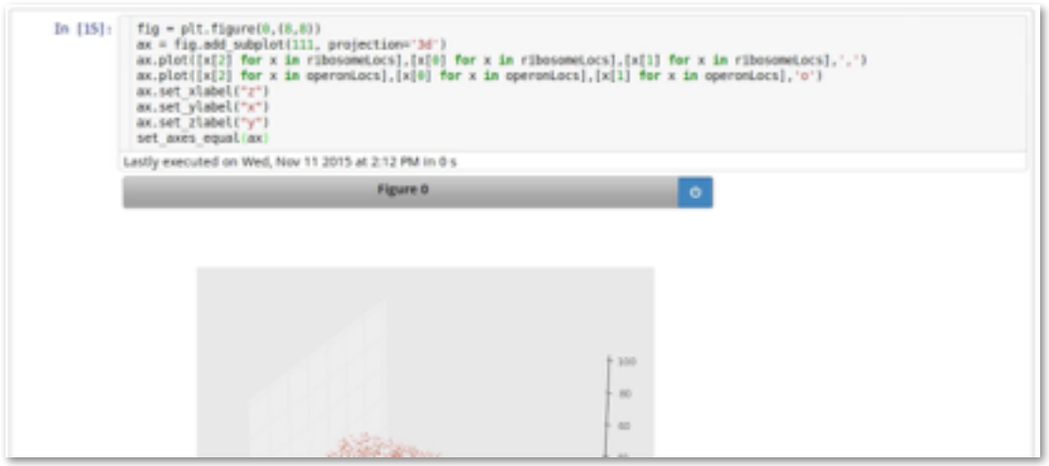
- Prerequisites to build from source:
  - CUDA (https://developer.nvidia.com/cuda-downloads)
  - HDF5 (www.hdfgroup.org)
  - Protocol Buffers (https://developers.google.com/protocol-buffers/)
- Optional:
  - Python (www.python.org)
    - and NumPy, SciPy, h5py, matplotlib, igraph….
  - libSBML (http://sbml.org/Software/libSBML)
  - VMD (http://www.ks.uiuc.edu/Research/vmd/)
  - MPI (https://www.open-mpi.org/)

LM In the Cloud

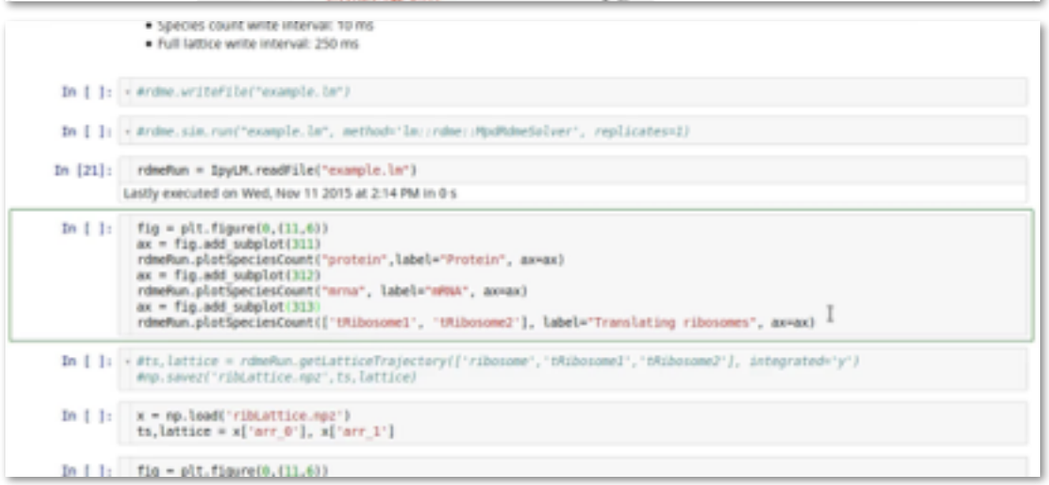
- Nothing to install: need only a web browser
- Up and running with an interactive notebook interface in minutes
- Pay as-you-go



Import experimental data



Design cell simulation

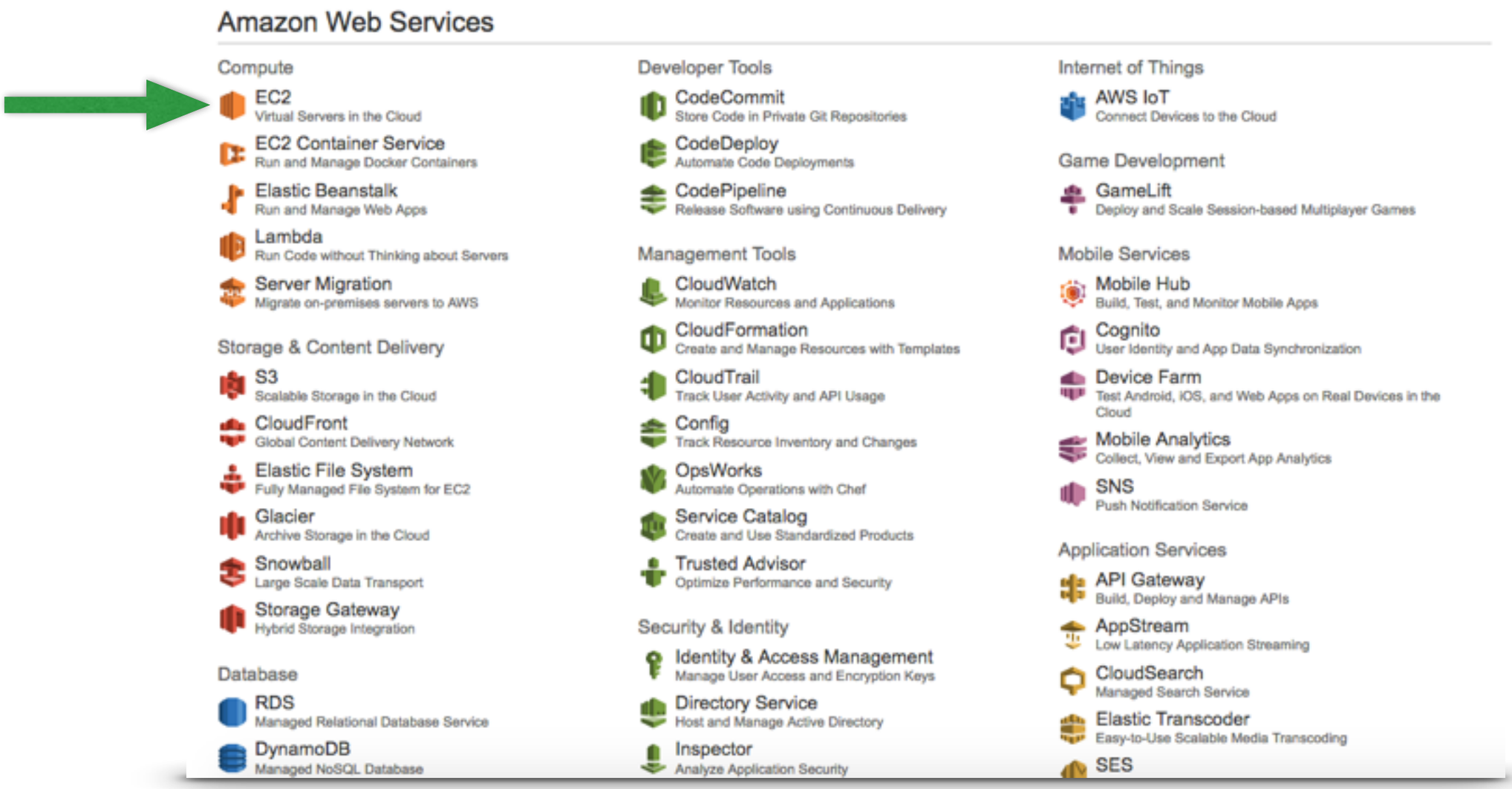


Analyze trajectories

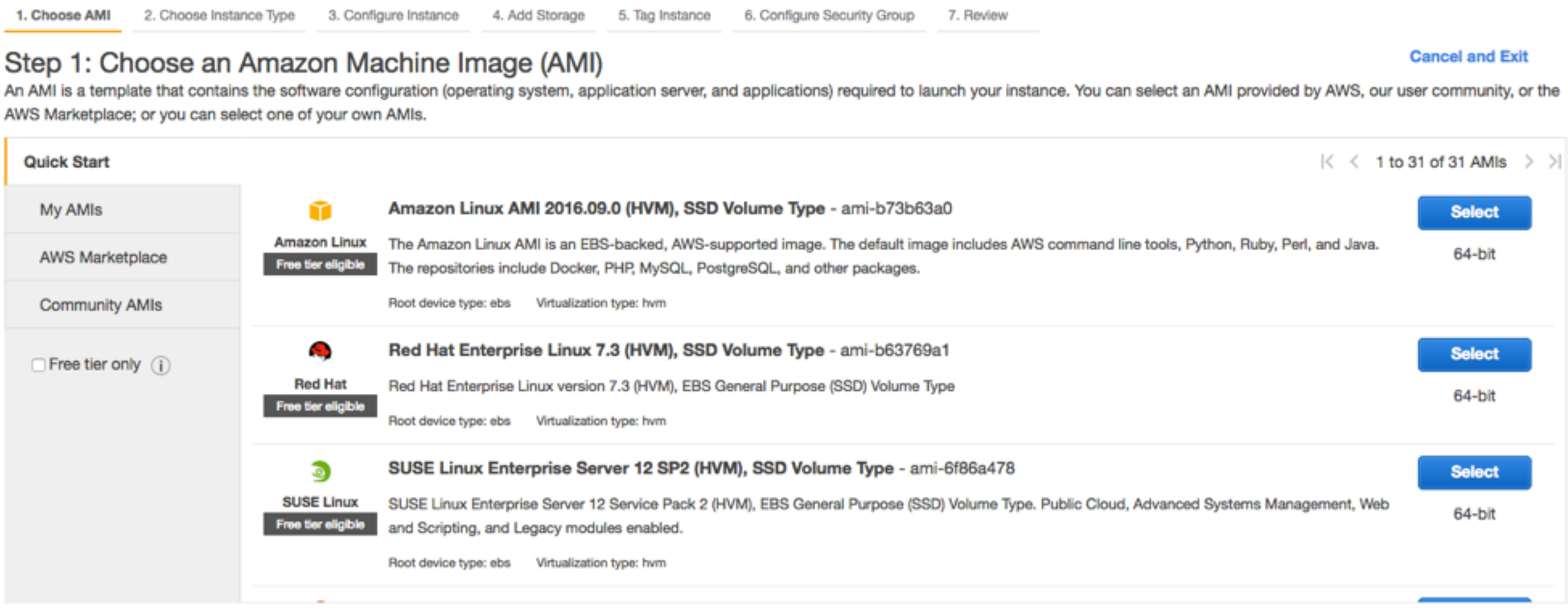
amazon web services™

AWS: Globally Distributed

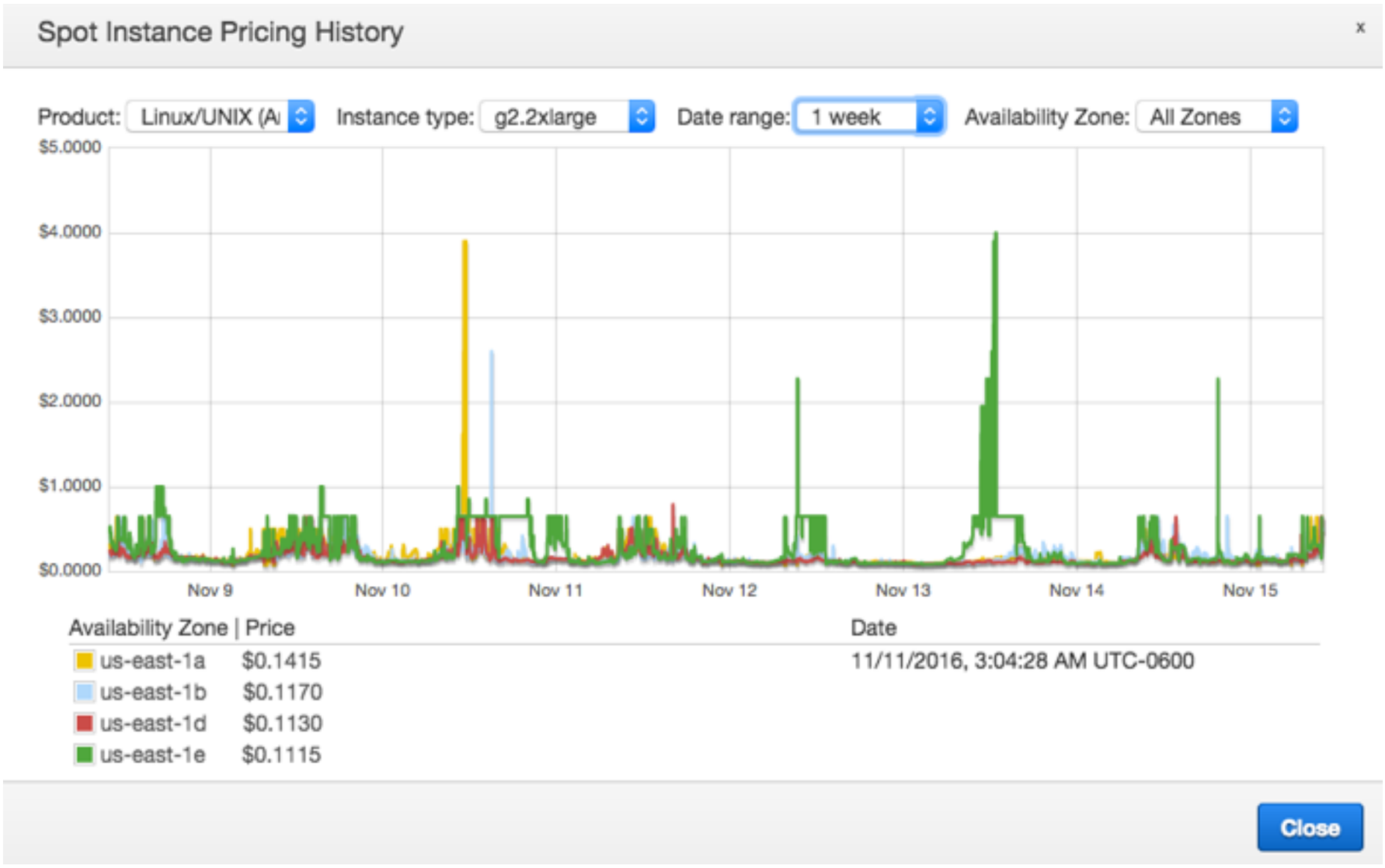
Wide array of services



Creating an Instance



What does it cost?



Jupyter Notebook

jupyter Tut1.2-StochBimol Last Checkpoint: 6 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Help Python 3

Markdown CellToolbar

Tutorial 1.2 - Stochastic Solution of a Bimolecular Reaction

Here we examine a stochastic version of Tutorial 1.1.

In Python you "import" libraries to be able to use their functionality. The first several lines import certain functionality including certain operating system functions (os), standard numeric capabilities that are much like Matlab (numpy) and plotting capabilities (matplotlib). There lines are boiler-plate code for most pyLM scripts.

In order to use pyLM we need to import several libraries. The first is pyLM proper (pyLM). The second is a library with a number of functions such as `nm()`, `micron()`, `ms()`, `microsecond()`, etc. that allow cleaner definition of units. Finally, we import the `pyLM` standard library of functionality `pySTDLM`, which contains standard plotting and post-processing commands.

In [ ]:

```
# Import Standard Python Libraries
import os
import numpy as np
#import matplotlib.pyplot as plt


# Import pyLM Libraries
from pyLM import *
from pyLM.units import *
from pySTDLM import *
from pySTDLM.PostProcessing import *

# Enable plotting inline in the Jupyter notebook
%matplotlib inline
```

Hit Shift+Enter to execute cell