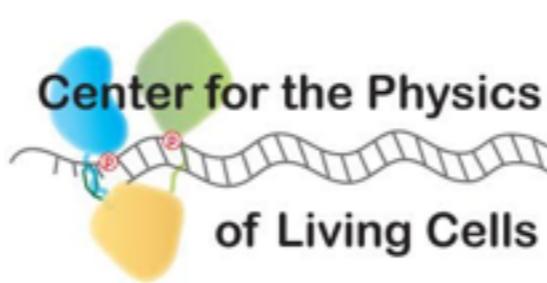
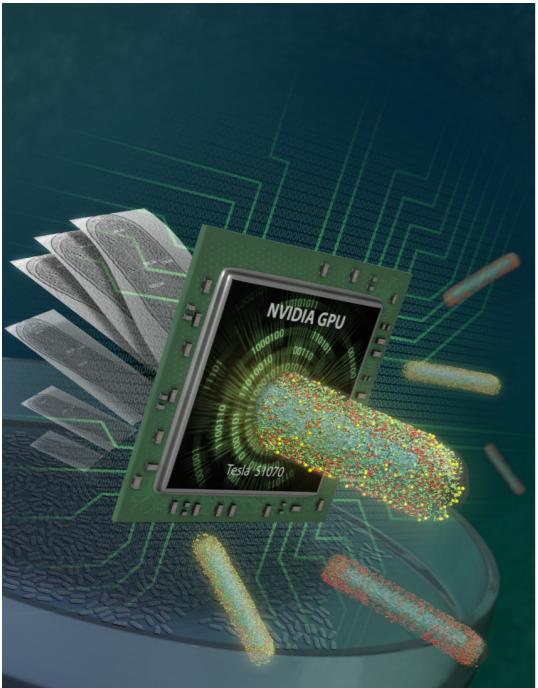


Part 2: Introduction to Using Lattice Microbes

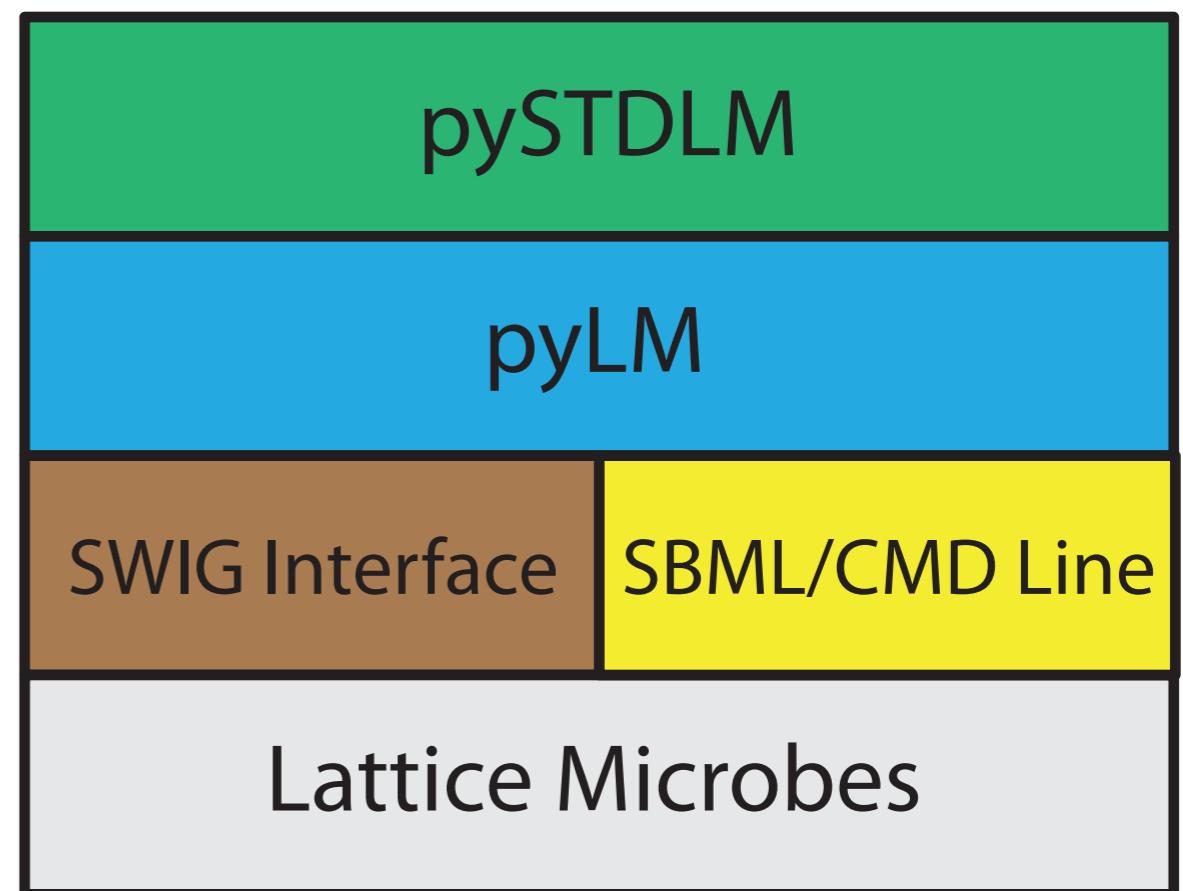
Joseph R. Peterson and Michael J. Hallock
Luthey-Schulten Laboratory
University of Illinois at Urbana-Champaign

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



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

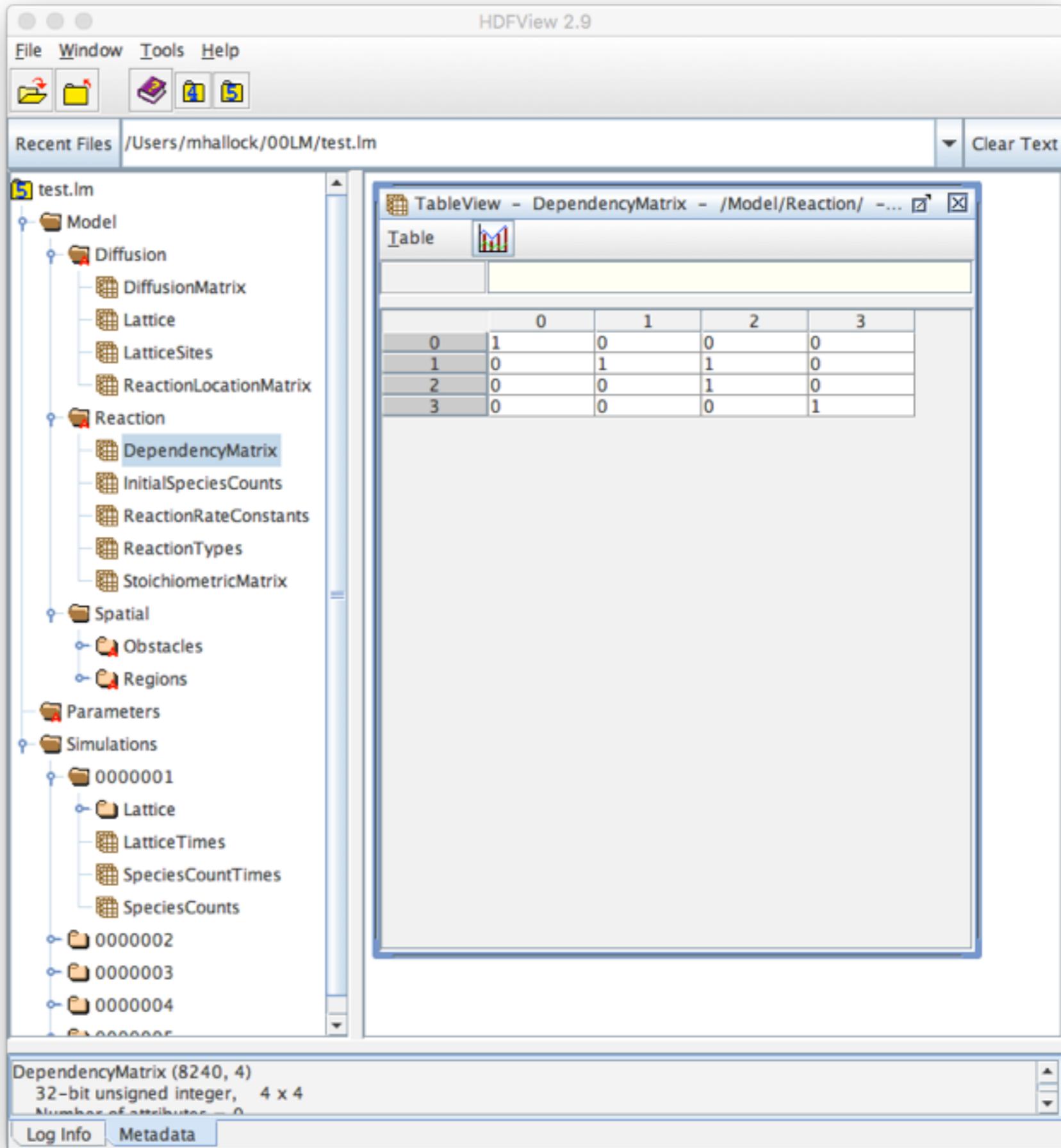
Spatially resolved (RDME)

```
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

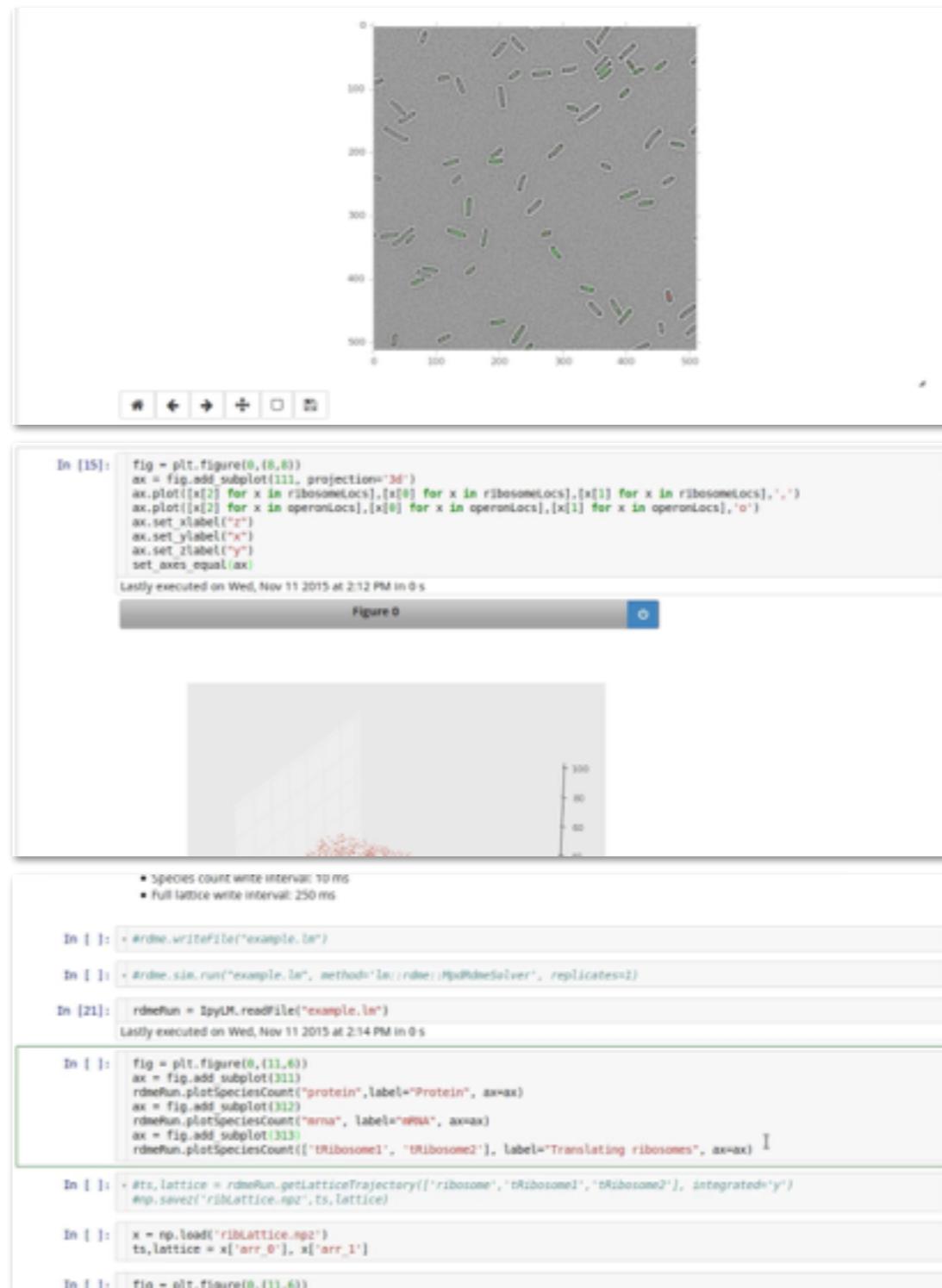


Installing LM

- 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

AWS: Globally Distributed



Wide array of services

Amazon Web Services			
Compute	Developer Tools	Internet of Things	
 EC2 Virtual Servers in the Cloud	 CodeCommit Store Code in Private Git Repositories	 AWS IoT Connect Devices to the Cloud	
 EC2 Container Service Run and Manage Docker Containers	 CodeDeploy Automate Code Deployments	 GameLift Deploy and Scale Session-based Multiplayer Games	
 Elastic Beanstalk Run and Manage Web Apps	 CodePipeline Release Software using Continuous Delivery	 Mobile Hub Build, Test, and Monitor Mobile Apps	
 Lambda Run Code without Thinking about Servers	 CloudWatch Monitor Resources and Applications	 Cognito User Identity and App Data Synchronization	
 Server Migration Migrate on-premises servers to AWS	 CloudFormation Create and Manage Resources with Templates	 Device Farm Test Android, iOS, and Web Apps on Real Devices in the Cloud	
Storage & Content Delivery	 CloudTrail Track User Activity and API Usage	 Mobile Analytics Collect, View and Export App Analytics	
 S3 Scalable Storage in the Cloud	 Config Track Resource Inventory and Changes	 SNS Push Notification Service	
 CloudFront Global Content Delivery Network	 OpsWorks Automate Operations with Chef	 Application Services	
 Elastic File System Fully Managed File System for EC2	 Service Catalog Create and Use Standardized Products	 API Gateway Build, Deploy and Manage APIs	
 Glacier Archive Storage in the Cloud	 Trusted Advisor Optimize Performance and Security	 AppStream Low Latency Application Streaming	
 Snowball Large Scale Data Transport	 Identity & Access Management Manage User Access and Encryption Keys	 CloudSearch Managed Search Service	
 Storage Gateway Hybrid Storage Integration	 Directory Service Host and Manage Active Directory	 Elastic Transcoder Easy-to-Use Scalable Media Transcoding	
Database	 Inspector Analyze Application Security	 SES	
 RDS Managed Relational Database Service			
 DynamoDB Managed NoSQL Database			



Creating an Instance

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

[Cancel and Exit](#)

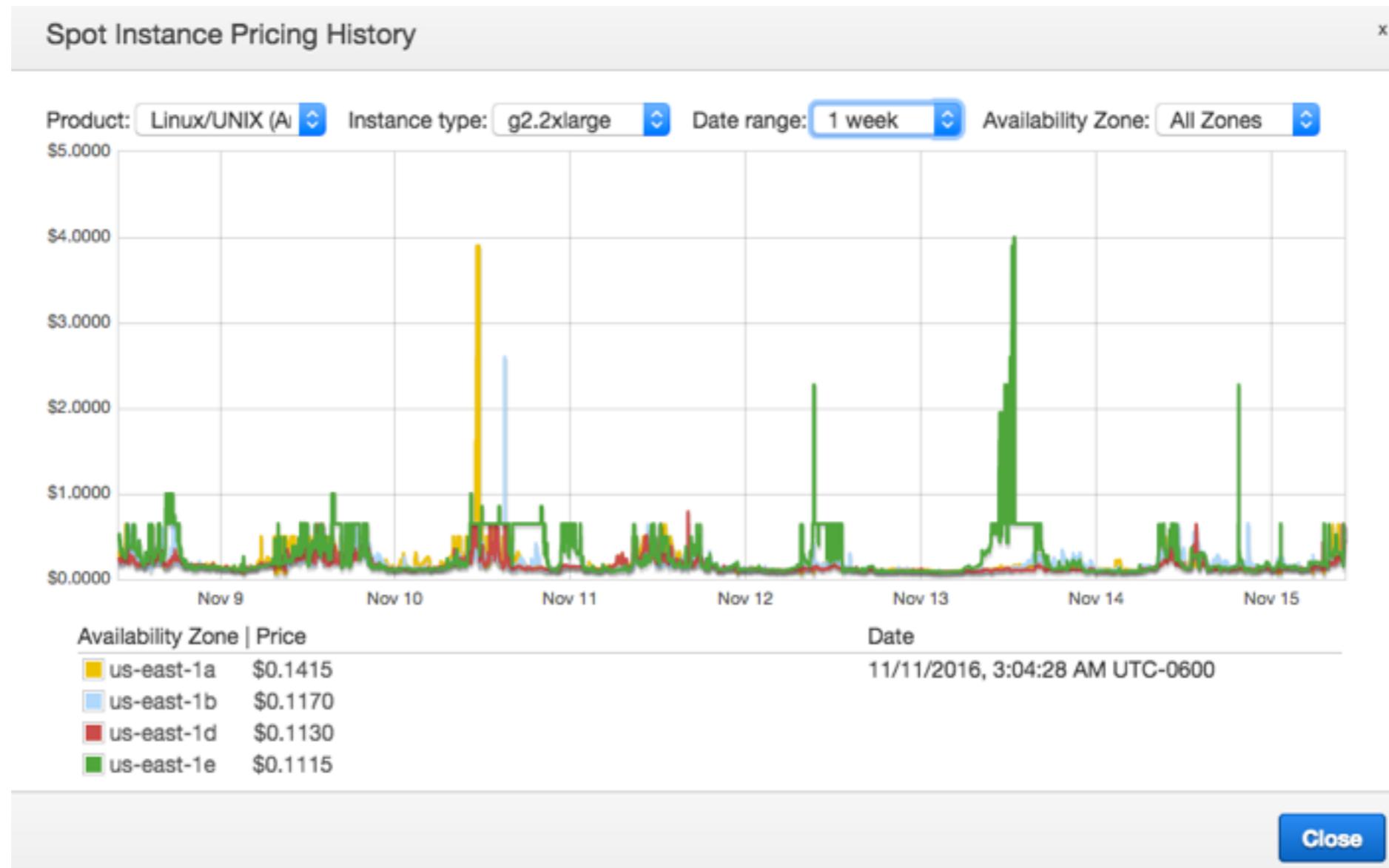
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start

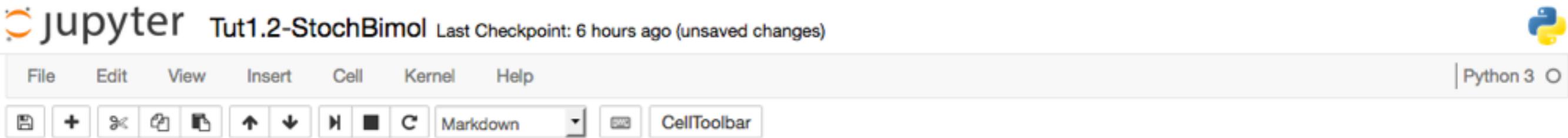
< < 1 to 31 of 31 AMIs > >

My AMIs	Amazon Linux AMI 2016.09.0 (HVM), SSD Volume Type - ami-b73b63a0	Select
AWS Marketplace	Amazon Linux <small>Free tier eligible</small> The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.	64-bit
Community AMIs	Root device type: ebs Virtualization type: hvm	
<input type="checkbox"/> Free tier only <small>(i)</small>	Red Hat Enterprise Linux 7.3 (HVM), SSD Volume Type - ami-b63769a1 Red Hat <small>Free tier eligible</small> Red Hat Enterprise Linux version 7.3 (HVM), EBS General Purpose (SSD) Volume Type	Select 64-bit
SUSE Linux <small>Free tier eligible</small>	SUSE Linux Enterprise Server 12 SP2 (HVM), SSD Volume Type - ami-6f86a478 SUSE Linux Enterprise Server 12 Service Pack 2 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.	Select 64-bit

What does it cost?



Jupyter Notebook



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). These 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 pySTDL, 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 pySTDL import *
from pySTDL.PostProcessing import *

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

Hit Shift+Enter to execute cell