Maxwell: bringing cloud-powered electromagnetic simulations to Matlab

Advanced user interface tutorial

- Quick-start
- Examples
- How Maxwell uses the cloud (EC2)
- How Maxwell solves electromagnetics

Jesse Lu, Jelena Vuckovic group, Stanford University

Definitions

- What is Maxwell?
  - a Matlab toolset
  - that uses Amazon's Elastic Compute Cloud (EC2)
  - to solve 3D frequency-domain electromagnetic simulations.

- Features:
  - Cryptographically-secure communication (https)
  - Full control over all simulation parameters
  - GPU-acceleration provided by Nvidia Tesla GPUs
  - Queueing system to allow for full usage of cluster
  - Scalable to hundreds of simultaneous simulations running on hundreds of nodes.

Maxwell provides two user interfaces: advanced and other

- advanced:

- other:

This presentation covers the advanced interface

Quick-start

- Sign up at

```
% Download maxwell.m
>> urlwrite('m.lightlabs.co', 'maxwell.m');

% Provide AWS credentials and launch a 2-node cluster.
>> maxwell.aws_credentials('aws-access-id', 'aws-secret-key');
>> maxwell.launch('cluster-name', 2);

% Run simulation on 1 node.
>> [E, H] = maxwell.solve('cluster-name', 1, ...);

% Terminate cluster
>> maxwell.terminate('cluster-name');
```

Wait, what just happened?

- `urlwrite()` downloaded the advanced interface for Maxwell,
- `maxwell.aws_credentials()` provided the AWS credentials that
- `maxwell.launch()` needed to create a cluster on EC2.
- `maxwell.solve()` solved the electromagnetic simulation on the cluster and downloaded the resulting electromagnetic fields, and
- `maxwell.terminate()` terminated the EC2 cluster.

Examples

How Maxwell uses the cloud (EC2)

- Maxwell uses your Amazon Web Services (AWS) account to
  - Create a custom Amazon EC2 cluster, and
  - Solve electromagnetic simulations on it;

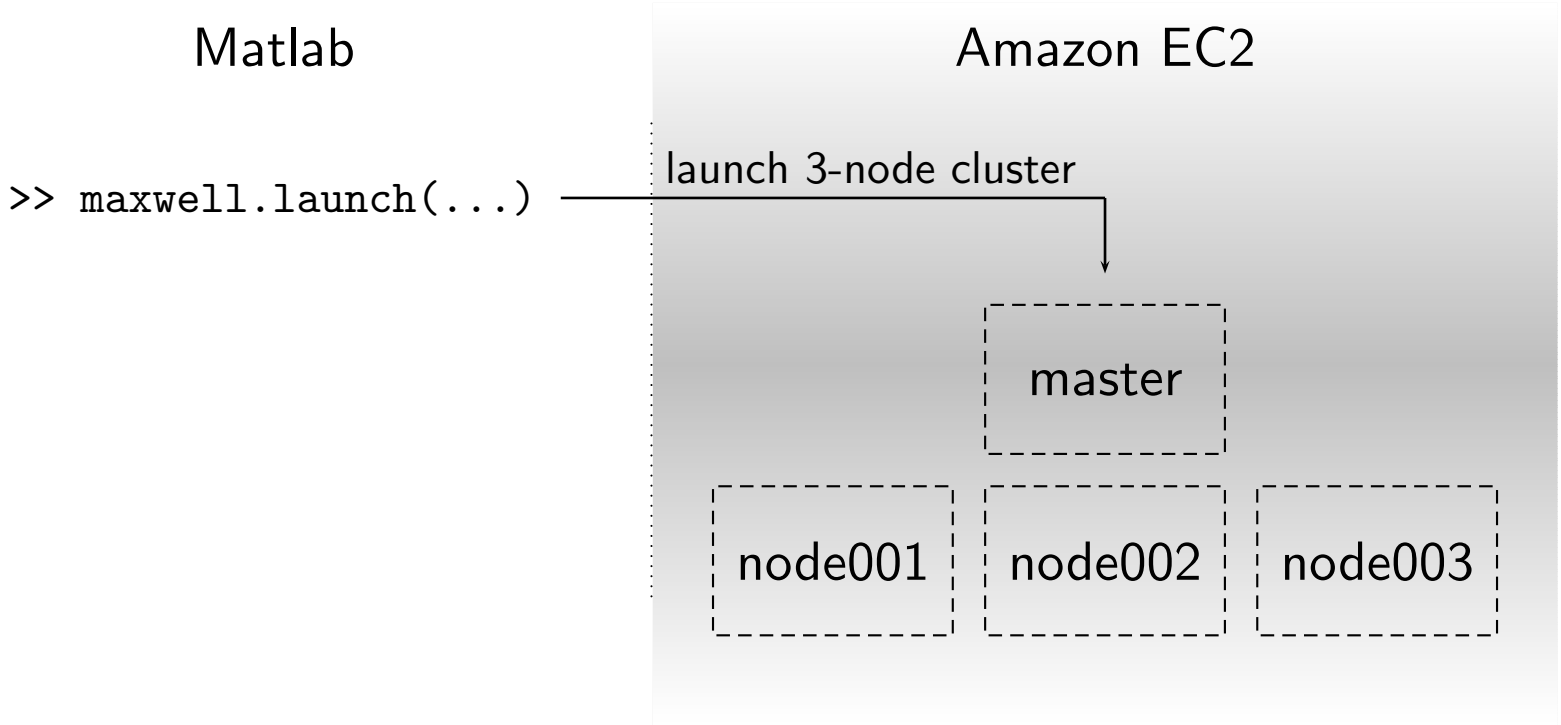
  all without leaving your local Matlab environment.

- To get started, you need to
  - sign up for an AWS account,
  - retrieve your AWS security credentials, and
  - purchase the custom Maxwell Amazon Machine Image (AMI).

  For detailed instructions see Website.

- Maxwell's advanced interface comprises of just five commands:
  - `maxwell.aws_credentials()`
  - `maxwell.launch()`
  - `maxwell.solve()`
  - `maxwell.solve_async()`
  - `maxwell.terminate()`

- `maxwell.aws_credentials('aws-key-id', 'aws-secret-key');`
  - Stores the security credentials linked to your AWS account locally
  - Security credentials are used to launch and terminate clusters
  - Transmitted over https and never stored on server-side
  - Tutorial on obtaining your credentials at Website.



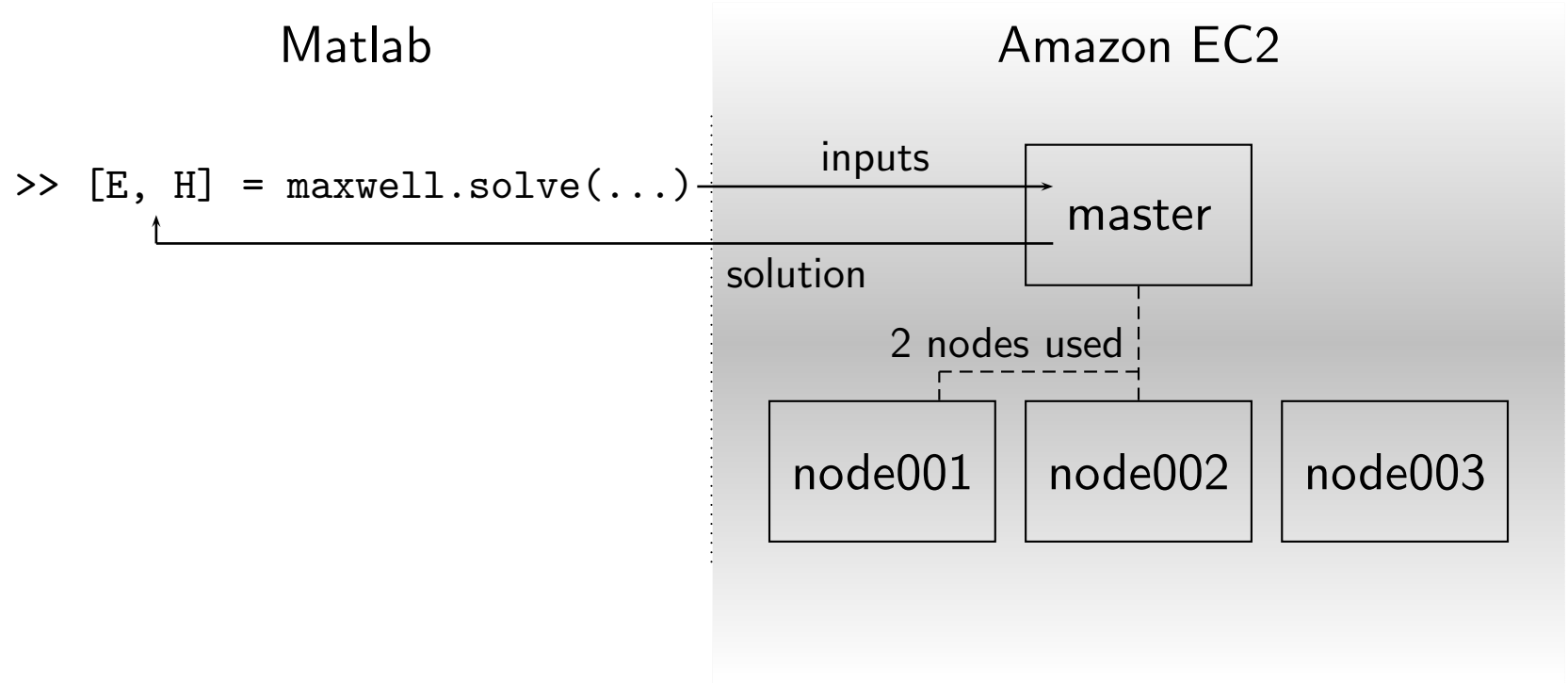
- `maxwell.launch('cluster-name', num_nodes);`
  - Creates an EC2 cluster consisting of 1 master node and `num_nodes` worker nodes

  - `'cluster-name'` parameter allows for using multiple clusters at once.
  - The launch can be monitored manually from the EC2 Management Console at `console.aws.amazon.com/ec2`

- The master node is launched
  - with the paid Maxwell Amazon Machine Image (AMI),
  - as an on-demand instance, and
  - as an `m1.medium` instance.

- The worker nodes are launched
  - using a public (free) AMI,
  - as spot request instances (in order to achieve up to 80% savings), and
  - as `cg1.4xlarge` instances.

Note that the use of spot requests for worker nodes may result in sudden cluster termination, in this case the cluster will need to be terminated and a new cluster should be started.

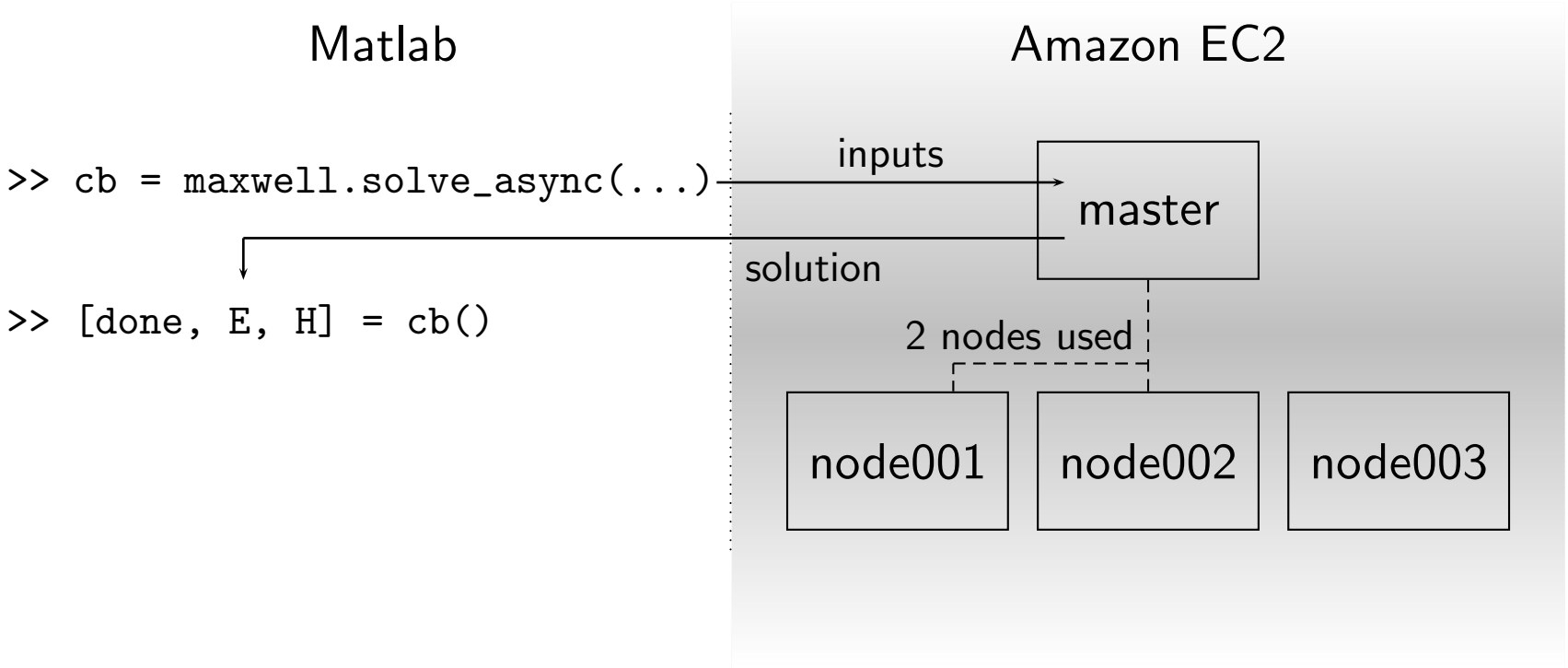


- `[E, H] = maxwell.solve('cluster-name', n, ...);`
  - Solves an electromagnetic simulation on `n` nodes of cluster `'cluster-name'`

  - Additional simulation parameters "..." described in following section
  - Returns as solution both electric and magnetic fields
  - For full documentation of this function see Website

- `maxwell.solve()` proceeds as follows:
  - Transfers simulation parameters to the specified cluster
  - Waits for worker nodes to be provisioned for the simulation
  - Continues to wait as simulation is executed on worker nodes
  - Retrieves simulation results back to Matlab

- Although attempting to use more nodes than available in the cluster will result in an error, the provided queueing system does allow for the *total* number of requested nodes to exceed the number of nodes in the cluster.



- `callback = maxwell.solve_async('cluster-name', n, ...);`
  - Asynchronous solve that returns a callback function instead of waiting for the simulation to complete

- The callback function is then used to check for solve completion and to retrieve the simulation results:

```
[is_finished, E, H] = callback();
```

- If the solve has not finished, `is_finished` returns `false` and `E` and `H` both return empty cell arrays.
- The additional simulation parameters "..." are identical to those used in `maxwell.solve()` and are detailed in the following section

- `maxwell.solve_async()` allows even single-threaded Matlab users to simultaneously execute a virtually unlimited number of simulations.
  - `maxwell.solve_async()` proceeds by uploading the simulation to the cluster and then immediately returns the function `callback`.

- `maxwell.terminate('cluster-name');`
  - Terminates the cluster `'cluster-name'`
  - Note that AWS instances are charged by the hour and that partial hours are charged the full hour.

How Maxwell solves electromagnetics

- In this section we detail the simulation parameters used by the `maxwell.solve()` and `maxwell.solve_async()` functions

- Along with the `'cluster-name'` and `n` parameters, both functions support the following parameters which describe the physical simulation:

  - `omega`
  - `d_prim, d_dual, s_prim, s_dual`
  - `mu, epsilon`
  - `E, J`
  - `max_iters, err_thresh`

  and can be understood from the master equation:

$$\nabla \times \mu^{-1} \nabla \times E - \omega^2 \epsilon E = -i\omega J \tag{1}$$

- Frequency parameter: `omega`
- Spatial grid parameters: `d_prim, d_dual, s_prim, s_dual`
- Material parameters: `mu, epsilon`
- Field parameters: `E, J`
- Convergence parameters: `max_iters, err_thresh`