Scripts for GCBM-FLINT Course Videos

Video-2: Slides- 9 to 15

  
  
10th Slide

As I mentioned before it makes the FLINT very scalable, but it does make certain problems very difficult for it to handle.

We cannot deal with awareness of neighboring pixels so we cannot work with fire spread, and we do not really know the exact location of rule-based disturbances.

11

This is a very high-level overview of what the event-driven system looks like. Each pixel is simulated through its whole timeseries before moving on to the next. The pixel data is loaded, then the sequencer module fires a sequence of system lifecycle events that science modules subscribe to in order to do their processing; for example, the timing init event notifies modules when a new pixel is loaded, then the timing events are looped over - timing step start is the beginning of the current timestep, timing step end is the end of the current timestep, the output event is for any post-timestep reporting, etc. (the events shown are only a subset of what actually happens)"

12

One of the examples of our modules is the growth module. It subscribes to three of those system events, the simulation start- loads the root biomass equation, each time a pixel is loaded, it loads dead organic matter turnover rates and the timings step is going to load or process the current growth curve.

13

CBM science is implemented as modules running on the front platform. The key features are that ecosystem components biomass soil greenhouse gases are represented as carbon pools. There is a yield curve-based growth module, and the disturbances and annual processes are represented as transfers between these carbon pools.

The core module includes   
- biomass growth and mortality  
- dead organic matter and soil dynamics and   
- disturbance impacts - management activities like harvest, natural disturbances like fire and land use changes like deforestation, afforestation.

14

We went through a pretty lengthy process when we developed this new model and the results are nearly identical to CBM CFS 3.

Let’s look at the main differences from CBM CFS3. forest in inventory and disturbance data come from spatial layers instead of database tables.

The location of all the disturbance events is explicit in spatial layers instead of using rules.

It gives out special as well as tabular output for pools and fluxes also it is easier to extend with new models and easier to simulate large landscapes such as the ones that have a huge number of pixels.

15

Here is an example of one of the outputs that we generated using the model. The simulation does not produce this directly but there is a set of advanced python scripts that we used to produce these outputs. They are also freely available along with the ones that generate animations using the spatial output.



16

We start off with some spatial data like disturbances and forest inventory data, along with a database of non-spatial modelling parameters like growth curves and disturbance matrices. This passes through a set of python based pre-processing tools to output data in GCBM readable format.

And the along with GCBM configuration files in JSON format go for GCBM simulation. It produces a set of spatial outputs as well a SQL database output.

17

Talking about the data that we need for running GCBM. On the spatial side, we need At least a map of the initial forest age or the time since last disturbance and the classifiers. It is recommended to have a mean annual temperature layer which affects the decay rates and reference parameters used are almost similar to that of CBM-CFS3. An administrative, ecological boundary is also used.

In terms of non-spatial or tabular data, it requires a CBM-CFS3 archive index database which contains the library of non-spatial modelling parameters and a CBM-CFS3 styled yield table in CSV or Excel format. Using a country specific CBM-CFS3 archive index database, that is customising it according to using the modelling parameters that are more suited to your geographical region improves the results.

18

Let's look at the archive index database. It contains the non-spatial parameters and spatially references parameters like the decay rates, disturbance matrices, and root biomass coefficients.

It is used in both CBM CFS3 and GCBM.

19

A little more detail on the disturbance events format for GCBM.

We can take pretty much any spatial layer format but usually  on the raster side it'll be a file per year where pixel value equals to the disturbance type or file per disturbance type and year where pixel value equals to 0 or 1, bits indicating disturbed or not disturbed.

We can also take in vector layers and have any number of those layers that contain polygons of completely disturbed areas. What you see on your simulation map is what ends up getting simulated. You will need a couple of attributes in the vector layers, like what's the disturbance year or what's the type of disturbance, etc.

And these will be paired with a lookup table that maps non-standard disturbance type names in the spatial layer to actual names of disturbance types that are understood by the model.

20

Getting into the structure of this training project, the standalone project is a collection of individual pre and post processing tools organised into a workflow to get people up and running with GCBM.

It does not do any sort of fancy pre-processing and is really what you see is what you get. It ingests SPATIAL layers and SIT like data and runs exactly what has been specified by the input data.

It is designed to be portable so you can copy it anywhere and run with correct python or GDAL environment

21

It is best suited for the project where the data already exists in the correct format and does not require any pre-processing before-hand.

I just not designed only for this training. All of our real-world projects basically use the same project structure and scripts.

The standalone template is the recommended way to get new projects up and running.

We do some advanced scripting internally, but this is the basic project structure used.

22.

This is the overview of standalone projects workflow. We start off with tiler that converts our raw data into GCBM readable format.

The next is a tool called recliner to GCBM that is responsible for creating the SQL database.

Once we've got the input data in place, we run the actual GCBM model itself. Once the run is finished, we get a couple of post processing tools. The first one is compiled GCBM spatial output, which stitches all the chunks of output to geotiff files. Then there's another tool- compile GCBM results that generates ecosystem indicators in to SQL like database

23.

Let's talk about the outputs given by our standalone project. It's usually a SQL like database, tabular form output that uses SQLite by default and postgres as well.

The spatial output is in envi or geotiff format. After post processing, it gets converted to the standard geotiff format.

And then it produces some flattened reporting tables that are more user friendly. They can be either SQLite or postgres databases.

24

There's a spreadsheet available describing exactly what goes into all those reporting tables. The reporting table format integrates easily with tools like PowerBI desktop, and we can create PowerBI templates to visualise results from different simulations. It has all the indicator names here on the left

25

This is a screenshot of one of the PowerBI templates to visualise the results.

26

The directory structure of the project template  looks like this.

Documentation folder contains some important materials, for example python snippets for configuring the tiler script, and basic assorted documentation.

Then we've GCBM folder That contains the simulation working directory

the next one is input Database that contains the a spatial input data for example archive index database , yield table and all

 next comes the layers which has the spatial layers inside it

Next we have a log files folder, processed output folder, and the tools that contain all the pre-and post processing tools GCBM itself and supporting software.

Then we have the license agreement for using GCBM, it is freely available to everyone.

Then we have the readme and batch file.

Project-Rectangle World: Slides- 27 to 30



Let's have a look at our sample standalone project named as rectangle world.

It has been set up with some exemplar simulation data. the initial python or GDAL environment should be set up for running this.

The spatial layers can be found in the layers folder in the raw subfolder and the yield table is stored in the input database directory as yield dot csv.

28

Let's start by having a look at the input database that we are going to use in this project. This is the inventory shape file.

This can be found in the standalone template folder in the layers subfolder as inventory dot SHP.

There are a couple of key pieces of information  that we need to add in the inventory layer.

Age 2010 is the forest age at the beginning of the simulation.

There are two classifiers for this.

In this project we will call them classifier one and classifier 2 but they can be named anything.

They link our  spatial data to the non-spatial yield table that describes how the forest grows.

29

The next layer in the sample data set is the disturbance layer.

Just like an inventory layer, you can find it in the standalone template folder, in layers subdirectory as disturbances dot SHP.

We have the first attribute as disturbance type which is already understandable looking at the values and then we have the year attribute.

30

Next we have got the non spatial data. This is what our yield table looks like.  This format is pretty similar to CBM CFS3.

This file can be found in the input database folder as yield dot csv.

Slides: 31 to 35



31  
Following the process for running a project in GCBM, the first thing that will do is configuring and running tiler to process the input spatial data into the GCBM format.

We will run the recliner to GCBM tool to create the GCBM input database.

There is a python script that will configure the GCBM for us.

Next we'll run the model.

And lastly we will run the post processing tools to generate the final output.

Familiar to users of CBM-CFS3, we generate all the usual ecosystem indicators like NPP in database tables.

And then we will convert the raw spatial output to final layers.

32

This slide gives us a basic technical background that we might need to understand some aspects of using GCBM.

Python is a general-purpose scripting language that we use for most of the pre and post processing work.

We will need a basic knowledge of python to run the GCBM.

There is a cheat sheet available in the documentation folder as tiler cheat sheet dot Txt that provides python snippets for the most common tiler use cases

The configuration files for GCBM are in JSON format that stands for JavaScript object notation

It is a common text file format that is used by many software packages.

GCBM and its supporting tools use JSON as a configuration file format.

33

Here are some prerequisites to run GCBM.

You can find the instructions in the readme dot Txt file.

if you have python 3.7 already installed on your machine, simply locate it by existing python installation.

Open a command prompt in tools slash python 3 installer and then type install modules only dot BAT.

In case python 3.7 is not already installed, open a command prompt in tools / python3 installer folder and type install python followed by the path of python.

Python will be installed into the specified path or into a directory called python 3 7 in the c drive by default.

34

since the archive index database is in MS access format so will need to install the access database engine.

and then we have some visual C++ redistributable packages and dot net 4.8 installers.

35

Once all of that is installed you can check that everything is running correctly just by double-clicking the run all batch file.

If everything goes well it's great but if not, here are some troubleshooting steps.

If the tiler script fails it's a python or GDAL issue. We will need to check the environment variables again. You can recheck that all the paths have been set properly and correctly.

If the recliner to GCBM fails its usual access database driver issue.

So you can try switching from the 64 bit version to the 32 bit one by changing the platform variable in the run all batch file. I'll show you the batch file here 😁

At the top there are some configuration lines we have platform variable which refers to access database. We can try changing its value here and hit the save button.

If the model fails to run and it's usually because one of our earlier tests failed.

if the post-processing script fails again it's usually one of those areas that we missed before, either because GCBM failed or a python or access database driver issue.

⏱So if everything runs properly you should see this on your screen when you run that run all batch file.

⏱⏱⏱⏱

We got past running the pre processing tools pretty early and this is actual GCBM running.

**Slides: 36 to 44**



36

Now let's start a guided tour of our template project.

Our first step is to configure tiler python script. We can feel that any type of layer vector or raster until that is supported by the GDAL library. These are a couple of links for the supported formats.

We have developed a python package called mojadata which is the library for converting the spatial layers and deal with resampling and reprojections, things like that.

We have a project template here.

you can find it on the given path as tiler dot py

⏱⏱⏱⏱⏱⏱⏱⏱⏱

Firstly will define the bounding box which defines the study area of our simulation.

 all the other spatial layers are cropped, re-projected and resampled to the bounding box.

 One of the important things to note about the bounding box is that we’re pointing to our inventory shape file. We can point to any spatial layer, but we'll sample using our inventory shape file.

We need to select an attribute out of it. We'll select which polygon to use for the simulation area. In this case, we've just selected the poly ID attribute where all of our polygons in the landscape have a value.

Next is the pixel size, basically the resolution that our simulation will run.

⏱⏱

These are a couple of recorded data layers here- initial forest age and here is the optional mean annual temperature layer.

Then we have the disturbance layers here- the type and year of disturbance.

The next couple of slides are just reference materials. They do not contain too much detail as of now but they can be referred to in the future if the need be.

37

Just remember that layers need to belong to one of these two classes raster and vector.

40

now let's have a look at the classifier layers

 That's what joins our forest inventory spatial data to the yield table.

As we saw before there are two classifiers in the spatial inventory layer as well as our yield table.

⏱⏱⏱

Let's go back to our actual tiler script. The classifier layers are here on the line number 56.

So we will define one of our vector layer classes.

 we can choose any name for them.

These names just need to match the classifier names that we define later on in our recliner to GCBM script.

Again, we are giving the paths of the shapefile and the attributes that we will select out of them are classified 1 and classifier 2.

Next we have classified tags so we can tag layers with user defined strings that are bits of metadata about our layers.

41

Our next step is to include any required or optional data that we have. So we have a couple of attributes like the initial age and mean annual temperature that we need to add here.

42

Next up we have the last step that is adding disturbance layers. The requirements for setting up the disturbance layer are that the year must be in four-digit format and that the value of disturbance type variable must match the disturbance type name in the project input database.

43

Now once we have set up the tiler script it can be run by itself using the batch file here called process spatial layers dot BAT

Alternatively, you can also run the run all batch file that runs all of the pre-processing tools, the simulation and the post processing together.

Once that's trip trance you should see the final GCBM format layers in layers tiled folder.

Here are two things that you will see in that output.

* A tiff file of the final cropped or re-projected version of each layer
* Json file for each tiff that contains the meta data and attribute table.
* The csv file called transition rules dot CSV that contains any transition rule data from the disturbance layers.
* And the study area JSON file that contains metadata about the tiled layers. the other scripts can also use this JSON file to configure the GCBM for us.

And you will also find the log file in the logs folder for the tiler script.

45

In this video, we’ll learn about generating the Input database.

And for that, we’ll use the Recliner 2GCBM tool. This is our only non Python data preparation tool.

It is written in C# and it’s bridging the gap between our old model and the new spatially explicit one. It takes in the old CBM3 style archive index database which is where all of our non- spatial modeling parameters are stored.

It takes in a yield table in csv or Excel format, and after a little bit of configuration it will generate the SQLite format GCBM input database. You can find the output in the input database folder and it will generate a log pile as well. This is the only log pile that actually ends up in the Tools folder instead of the logs one.

….

46

So again, as already mentioned before, it needs and archive index database and an yield table, and the yield curves for this sample dataset are stored in yield.csv file.

This is how the yield.csv file looks like: we have these two classifiers, they map to an archive index database for species type and then we have increments. Increments are volumes of woody biomass with 10 year age increments basically just describing how our forest in going to grow.

….

47

Before you start configuring the input database you need to make note of the classifier layer names in the Tiler script that we set up. So the names are totally under user’s control, they just need to be consistent between the tiler and the Recliner2GCBM tool.

….

48

To start up the tool, we are gonna go to our Standalone Template input database folder and there is batch file in there called “run\_recliner2gcbm\_gui.bat”. So just double click and open that

⏱️⏱️ and it should launch the tool. ⏱️⏱️

And now we can just follow along with these slides.

Basically the first thing to do is step one is to  browse to our achieve index database

⏱️⏱️ and again that’s in Standalone Template input database as “Archive\_Beta\_Install.mdb” ⏱️⏱️ and the next step is to specify the output database, so this is the one that is going to be generated at the end of our process here ⏱️⏱️ and we have browsed again and basically we just want to overwrite the one that’s already there. So input database folder and “gcbm\_input.db”, just hit Save and yes, we do want to replace it.

⏱️⏱️

 ….

49

And then we can hit Next to go to the next screen, and here is where we set up our classifier layers. Click the Add button, we will have to do that twice ⏱️⏱️

 ….

50

So there’s a few things to set up. The first thing to enter the classifier name in that top text box and these are the names that need to be the same as the tiler script. ⏱️⏱️ Let’s have a look at the tiler script again⏱️⏱️ Look at the classifier layers at line 57, the name of these classifiers, here Classifier 1, can be copied and pasted the names directly from there or you’ll be required to  remember how you spelled it, it’s case sensitive. So, we  entered Classifier 1 in there.

 We are going to browse to our yields table for step 2. ⏱️⏱️ So. that’s our yield.csv in the input database folder ⏱️⏱️ and you can see it populated with the csv file contents. We need to tell it which column has all the Classifier 1 values in it.

So we hit Select and then just click anywhere in that column A ⏱️⏱️ and our last step here is just to specify whether or not our input data has a header row (it should be checked up by default). In this case, we do have a header row so just make sure that is checked up there and then hit ok. And then we do pretty much the same thing for our Classifier 2 ⏱️⏱️

Browse to the yield table again, this time our values are in column B here and hit Ok. ⏱️⏱️ Now that we have furnished information for our Classifiers, we can just hit Next.

 ⏱️⏱️

 ….

52

So now we need to set up our growth curves (yield curves) and again this is a set of volume data describing tree growth. It’s linked to our spatial inventory using these two Classifiers ⏱️⏱️

 ….

53

Here, the first thing to do is tell it which file has our yield curves in it. Just hit the Browse button up in the top, go to our input database folder, and open the yield.csv again. You’ll see the data populated in the two Classifiers that we have just set up, in this left pane. Again, we have a header row in our csv file, so make sure you check up that box, and then we’ll start with the Classifier mapping.

The reason this step seems a little bit repetitive is because sometimes our list of all possible Classifier values can come from a completely separate file. In this case it's all coming from the yield table that we have, as a separate file named classifiervalues.csv

So we need to mention which column each Classifier maps to. Hit Select, Classifier 1, in column A again, and then Classifier 2, again in column B.

Our growth curve interval is the number of years between increments in this file. Generally, if you’re the science person responsible for setting up a yield table, you’ll know what your growth curve interval is. In this sample dataset it’s 10 years. We kind of give a hint about that with the header row column series, so it’s counted by tens. We have to enter that in the box.

We need to tell it which column is the species, so that’s in the archive index tree species. In this case it’s column C. Generally, by convention, it’s in between the Classifier and the increments. Then finally we need to tell it which columns are the start and end of our yield table increments. So again click the ‘...’ . Our increments start in column D, so click anywhere in there. Then scroll all the way to the right and tell it which column our increments end on and then hit Next.

 ….

54

Transition Rules basically describe what happens after certain disturbances events. These are linked and created by that tiler script. In the sample dataset, we didn’t really go into a lot of details about it, but we have one very simple transition rule. In our tiler script, it’s on the line number 88 to 91.

This is optional in disturbance events, but we have one here. Basically, it just resets the stand age zero after a disturbance ⏱️⏱️ so we need to tell our tool about this file, just like the yield curves. Again, we’ll Browse to the transition rules file, and that is stored in the Layers folder, tiled, along with all of our data here. So it’s this transition\_rules.csv file. Again make sure the header row is checked up. And there’s a lot going on in this window. Some of these are advanced features and they’re optional.

For a basic transition rule, we set up use of transition classifiers. We do not need to get into a lot of details and how transition rules work, we can just focus on the mapping for now ⏱️⏱️ So again, we’re just finding the classifier columns in the data. Name is just this ID column here. Could be a string but in this case it’s just a numerical code reach transition rule. Regen delay is column B. Reset age is column C. And that’s all we have to do for the basic transition rules.

⏱️⏱️

 ….

55

This screen it’s optional. It’s one of the more advanced GCBM module things, so we just skip over it for now.  ⏱️⏱️

 ….

56

Now we’ve basically set up our input database. You can always go back and make changes if you need to. First thing you might want to do on this screen is to Save the Configuration, in case something goes wrong. It also will allow us to regenerate the input database from the command line if we save our configuration. So when we do that runall.bat file, that’s what it’s doing. So it’s taking a previously saved configuration and regenerating the input database. That allows this to play around with the values and the yield table or add extra yield curves and things like that, and we can just regenerate the database without having to do all that configuration over again. ⏱️⏱️

And hence we’ll just overwrite that existing configuration in there ⏱️⏱️ once you’ve done that, you can hit the Load button, and if everything goes well, everything was set up correctly, you should see the progress bar with no errors ⏱️⏱️

 ….

57 and 58 - Both skipped by Max

 ….

59

So now we can just hit Done out of that tool.

….

60

So now we have all our input databases spatial input processed. Now I can go ahead and run the model.

The raw output all got stored in the gcbm\_project folder, so we can see our configuration files in here that control the simulation. There is raw output in here. Our configuration templates are in here still if you want to change any of the simulation parameters. We basically added the template and then the actual active config files can generate from these templates.

⏱️⏱️

 ….

61

So we have a few different config files from the model. The most important one is the Provider configuration, which defines all the data sources that can be referenced by the modules and variables and things. The file paths in there can be either absolute or relative to the directory from wherever you run GCBM from. And then we have our Project config files that define everything about the simulations, so the modules to run, the order they run in, all of the carbon pool names in the simulation

From 1:50 until 4:40 ⏱

….

62

The Project config files got a few that are mainly for internal use that aren’t usually modified by the user, so it’s just an internal variable usage file. The modular configuration, the main thing to know about that one is you can go in there and disable modules if you run into any errors and things and you want to kind of narrow it down to a specific module that it’s going wrong. The carbon pools aren’t generally changed around it all, because it depends on what modules you’re running.

There is a spinup.json file which is basically how we initialize the carbon pools at the beginning of a simulation. And that can also by disabled from the config file for debugging. ⏱⏱

….

63

A couple of them are a little more user-modifiable. It can change the start and end date of the simulation in the localdomain.json config file. You can also tune the number of threads that the model is using. Usually you want to match the numbers of cores in your machine up to a maxim of about 10, otherwise it starts slowing down.

You can edit the output modules and this is useful if you want to be a little more selective about your spatial output, because they take a lot of disk space. But you can also define your ecosystem indicators so you can choose which carbon pools you want to edit together, which fluxes you want to report on, so it’s really flexible there.

And there’s a variables’ configuration file that points to the data providers. It has a basically default setup, so certain things that expects to have such as spatial layers and database queries to get things out of the database for, but if you want you can basically swap up a database query for a spatial layer or the other way around, so you can change up where you store your data.

….

64

But generally in this project framework, we don’t often have to edit those config files by hand. A lot of it it’s taken by care scripts and the bat files that we use. But if you do need to edit the templates, you can find them in the GCBM project templates folder again ⏱️⏱️ and for people who aren’t really familiar with editing json, after you make changes, there is a handy website tool to validate the json. ⏱️⏱️

….

65

Again, those are the most common things to edit. So we’ll start with the dates and the number of threads. And in the output configuration you can disable and reenable the spatial outputs ⏱⏱ so just to give you a little idea of what that looks like: this is the modules output.json file. This is our spatial output module and we defined a whole pile of different spatial outputs here, so this one here is outputting all of the aboveground biomass carbon by adding up these six pools defined here. You see you can enable and disable individual spatial outputs.

⏱️⏱️

….

66

There’s another bat file in the GCBM project called ‘update\_gcbm\_configuration.bat’. You can run that by itself, just to get all the configuration up-to-date. That’s also part of the ‘run\_all.bat’ file that we’ve been using so far. Or you can also just run it yourself on the command line.

⏱️⏱️

….

67

So I already showed you the model running. But if you want it to run just the GCBM part you can use the ‘run\_gcbm.bat’ file there or again you can just use that ‘run\_all.bat’ file ⏱️⏱️ once the model has run, the final couple of steps are to run the post processing tools. So there’s CompileGCBMResults, which turns the raw database output into more familiar, more user-friendly ecosystem indicators. You can compose indicators from individual pools and fluxes by editing CompileGCBMResults.ini

Again, you can run that by itself or it’s all part of the run\_all.bat file ⏱️⏱️ it will create a final output database in processed\_output. It’s in SQLite format and I’ll just show you quickly what it looks like: ⏱️⏱️

So we got the simulation year, the classifiers set, age pinning system, the area in hectares ⏱️⏱️ then we have all these different ecosystem indicators, so this is total ecosystem carbon by classifiers set by simulation year and age. You can see area, carbon numbers and carbon density.

….

68

So the ‘v\_flux…’ tables are the ones the user should be looking at. For people who aren’t really familiar with SQL, here are a few basic queries just to see what indicators are available in there. ⏱️⏱️

….

68

The units in those output tables are in hectares and tons of carbon, unless the column name says otherwise. And they give this state of the location at the end of the timestep. So after all the disturbances and growth that happened for that year.  ⏱️⏱️

….

70

Just a couple of observations on how the outputs are aggregated: so it’s by classifiers sets, in an age bin, at a particular year of the simulation. ⏱️⏱️ All the areas you see are for the entire classifier set age bin combination, so when you see disturbance numbers in there it’s not necessarily just the disturbed area, it’s the entire classifier set.  ⏱️⏱️

….

71

Then our last Post Processing tools that compile GCBM’s spatial output, so our landscape get’s  chunked up into these 100 little blocks of processing work. Each one of those it’s outputting a chunk of the landscape in terms of spatial output. This python script basically takes all these little chunks and comes up with single raster files. So we see that in our ‘processed\_output’. Here it’s all of our final whole landscape spatial output. ⏱️⏱️

….

72

And that’s just an example of one of the spatial outputs and it’s showing that it has run through the model and we’ve gotten our output. ⏱️⏱️

….

73

Just some Troubleshooting tips: we can find most of our tool simulation logs in the logs directory. ⏱️⏱️ So we have got a tiler postprocessing tools and update\_gcbm\_config, model debugging file. ⏱️⏱️⏱

It’s a good idea to check the Moja\_Debug.log, just to make sure there are no errors in any of the land units ⏱️⏱️ And you’re always kind of looking for earliest instance of errors, because it sort of cascade into everything else down the line. First thing to check it’s the tiler\_log.txt, and then check the update\_gcbm\_config.log and finally look through the model log for errors. If you’re lucky, sometimes you’ll also get some debugging information in the output database. ⏱️⏱️

….

74

For GCBM run time errors, if you don’t see any obvious errors in the data preparation tools, you can try stabling the modules until you isolate the exact one that is going wrong. You can do that by editing the module config file in the templates folder and then just keep rerunning GCBM. ⏱️⏱️

….

75

Just a couple of tips. You can configure the numbers of threads to use in the config file. Try and match the numbers of cores in the machine you are running on, but don’t go more than 10.

Start small or lower resolution, so configure the bounding box in the tiler script with something fairly coarse like 0.01, when you’re first getting a project up and running and then you can always rerun at a finer resolution later.

And, if you are running really big projects, you can always scale up to cloud computing or multiple machines. So right now, we’re doing most of our big simulations across a few hundred courses worth of cloud computing infrastructure.  ⏱️⏱️

….

76

This is just some advanced setup if you want to avoid having to run those bat files over and over again. ⏱⏱

I might actually get rid of these slides, because they are not quite as relevant anymore.

….