A Sustainable Collaboratory for Coastal Resilience Research

--Shuai Yuan, Steven R. Brandt, Qin Chen, Ling Zhu, and Rion Dooley





Outline

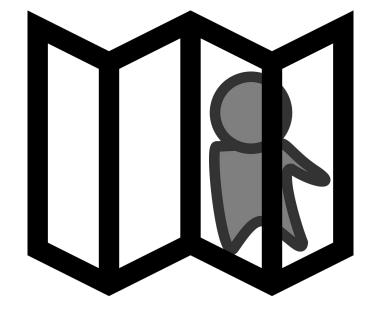
Introduction

Methodology

Workflow

System

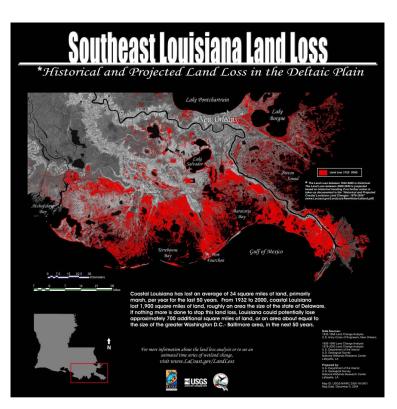
Result



Conclusion

Introduction

Problem: Communities on modern river deltas are threatened due to land subsidence, global reductions in river sediment, and rising sea levels. It is a grand challenge for earth system science to address these issues.

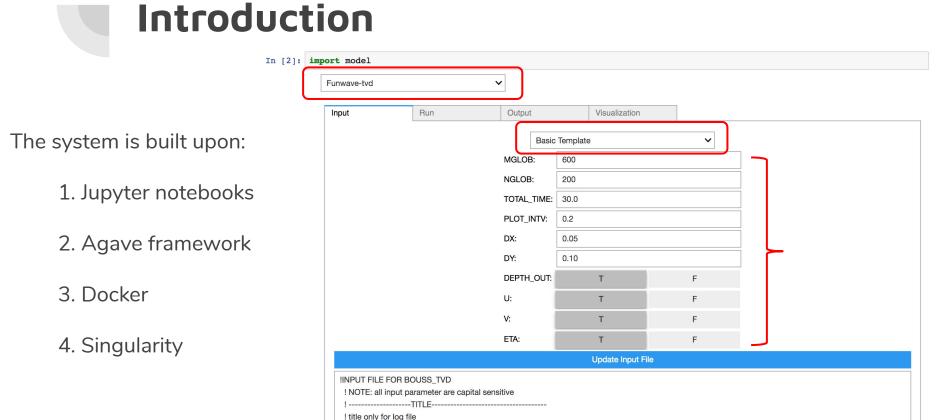


Example: Projected Mississippi River Delta by 2050

Introduction

Goal: Create a cloud-ready repository of open-source coastal modeling tools which enable scientists and engineers to more easily use high performance computers (HPC) and to study a variety of physical and ecological processes.

We want our system to be *intuitive*, *repeatable*, *collaborative*



TITLE = TEST RUN

! -----HOT START-----

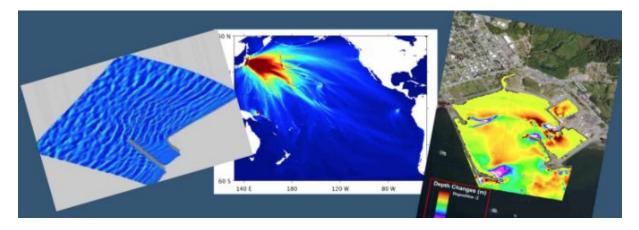
Introduction

<u>Swan</u>: A third-generation wave model, developed at Delft University of Technology, that computes random, short-crested wind-generated waves in coastal regions and inland waters

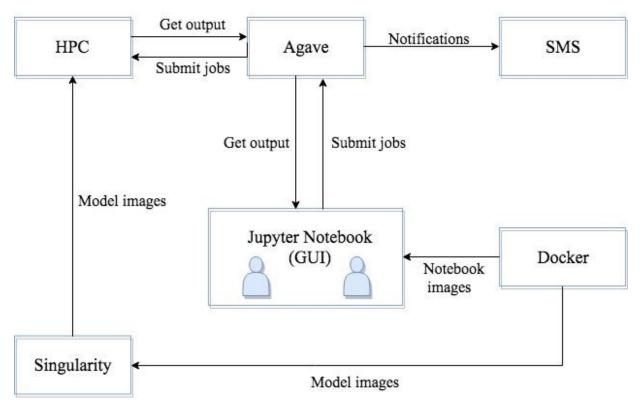


Introduction

<u>Funwave-tvd</u>: A Boussinesq model for simulating nearshore surface-waves, currents and tsunamis from ocean-basin to nearshore scales.



Workflow



Methodology--Docker and Singularity

 <u>Docker</u> is an open platform allowing developers to build, ship, and run distributed applications in self-contained environments.

Docker Hub makes it easy to share and discover images.

Security problem, not available on HPC resources.

2. <u>Singularity</u> makes it easy to use with MPI, and addresses the security concerns.

Methodology--Docker and Singularity

 Docker both to construct the images for singularity and to deploy the Jupyter notebooks to use machines and workstations

2. Singularity to deploy and run simulation codes on cloud-enabled resources

Methodology--Jupyter notebook

1. <u>Jupyter</u> is a tool for interactive computing that is halfway between a GUI and a command-line tool

2. Create a customizable, interactive environment for science discovery and engineering analyses

Methodology--Agave platform



THE LEADING ALL-IN-ONE SCIENCE-AS-A-SERVICE PLATFORM FOR THE OPEN SCIENCE COMMUNITY.



Methodology--Agave platform

1. <u>Agave</u>: restful JSON API

Abstracts a way the details of the job submission process

2. Provides a detailed provenance trail.

3. Fine-grained access controls for sharing jobs and data across machine boundaries.

System--before using

- Configure machine/resource
- Configure user access
- Permissions need to be given



System--configuration

In [1]:	import configuratio	n2	<pre>1 from agave import * 2 from ipywidgets import Box, Text, Layout, Label, Button 3 4 readpass("AGAVE PASSWD")</pre>			
	Password or secre Reading file `AGA Password or secre Reading file `PBT	VE_PASSWD.txt' et: PBTOK	<pre>5 readpass("PBTOK") 6 7 item_layout = Layout(8 display = 'flex', 9 flex_flow = 'row', 10 justify_content = 'flex-start', 11 width = '50%' 12) 13 14 agaveText = Text() 15 execMachineText = Text(value="shelob-exec-stevenrbrandt2")</pre>			
	Agave username Execute machine Storage system name Project name		<pre>16 machineNameText = Text(value="shelob-storage-stevenrbrandt23") 17 appText = Text(value="crcollaboratory-shelob-stevenrbrandt2-2.0") 18 pbtoRtext = Text() 19 confBtn = Button(description='Configure', layout= Layout(</pre>			
		shelob-exec-stevenrbrandt2	<pre>20 display = 'flex', 21 flex_flow = 'row', 22 justify_content = 'center', 23 width = 'l00px',</pre>			
		shelob-storage-stevenrbrandt23	24 disabled=False 25)) 26 27 def confBtn_click(a): 28 configure2(aqaveText.value, execMachineText.value,			
		crcollaboratory-shelob-stevenrbrandt2-2.0	29 machineNameText.value, appText.value) 30 31 confBtn.on_click(confBtn_click)			
	Configure					

System--input

<pre>import model</pre>					199 ####################################
Funwave-tvd	~				<pre>202 fwInputdd=Dropdown(options=['Choose Input Template', 'Basic Template'], value='C 203</pre>
					204 205 206 inputBox = Box(layout = Layout(flex flow = 'column'))
Input Run	Output		Visualization		207
					<pre>208 def fw_on_change(change): 209 inputTmp = ''</pre>
		Basic Templat	te	~	200 inputing = 7 210 items = []
		Date of templat			211 if change['type'] == 'change' and change['name'] == 'value':
	MGLOB:	: 600			<pre>212 if(change['new'] == 'Choose Input Template'):</pre>
					213 items=[]
	NGLOB:	200			214 inputBox.children = items
	NGLOB.	200			215 return
	TOTAL_1	TIME: 30.0			<pre>216 if(change['new'] == 'Basic Template'): 217 cmd("tar -zxvf input_funwave.tgz")</pre>
	TOTAL_I	11IVIE. 30.0			217 CmG(tai = 2x0 input input _ unwave.cgz) 218 input mp = 'input funwave/basic_template.txt'
	DI OT IN	NTV: 0.2			219
	PLOT_IN	NTV: 0.2			220 with open(inputTmp,"r") as fd:
	DY	0.05			<pre>221 for line in fd.readlines():</pre>
	DX:	0.05			<pre>222 g = re.search(r'(\w+)\s*=\s*\\${(.*)}',line) 223 if g:</pre>
	EV.	0.40			223 if g: 224 for match in re.findall(r'(\w+)=("[^"]*" \'[^\']* [^,\n]*)'
	DY:	0.10			<pre>if match[0] == 'value':</pre>
					<pre>226 label = line.split()[0]+'Label'</pre>
	DEPTH_	OUT:	Т	F	227 label = Label(value = line.split()[0].upper()+":")
					228 text = line.split()[0]
	U:		т	F	229
					230 box = Interspirit()[0] box = Layout(width = '10
	V:		т	F	232 items.append(box)
					<pre>233 if match[0] == 'option':</pre>
	ETA:	_	Т	F	<pre>234 label = line.split()[0]+'Label'</pre>
					<pre>235 label = Label(value = line.split()[0].upper()+":") 236 togBtns = line.split()[0]</pre>
			Update Input File		236 togBtns = line.split()[0] 237 togBtns = ToggleButtons(options=['T', 'F'])
					$238 \qquad box = line.split()[0]+!box'$
INPUT FILE FOR BOUSS					<pre>239 box = Box([label, togBtns], layout = Layout(width =</pre>
					240 items.append(box)
! NOTE: all input paramet	and the second sec				241
!TITLE-					242 inputBox.children = items 243
! title only for log file					243 244 fwInputdd.observe(fw_on_change) 245
TITLE = TEST RUN					245 246 def fwUpInput_btn_clicked(a):
!HOT ST	ART				240 del implificación interes (d): 241 input mp = ''
LOT CTADT - E					248 if(fwInputdd.value == 'Basic Template'):

System--customize input

```
! -----DIMENSION-----
! global grid dimension
Mglob = ${value=601,default=601}
Nglob = ${value=201,default=201}
```

	Basic	Template	~			
MGLO	3:	601				
NGLOB:		201				
TOTAL_TIME:		30.0				
PLOT_INTV:		0.2				
DX:		0.05				
DY:		0.10				
DEPTH	OUT:	т	F			
U:		т	F			
V :		т	F			
ETA:	1	т	F			

System--run

					285	######################################		
					286	<pre>run_item_layout = Layout(</pre>		
					287	display = 'flex',		
					288	<pre>flex_flow = 'row',</pre>		
					289	<pre>justify_content = 'flex-start',</pre>		
					290	width = '50%'		
					291			
					292			
	a second s				293	<pre>numnodeSlider = IntSlider(value=0, min=1, max=8, step=1)</pre>		
In [2]:	import model				294	<pre>numprocSlider = IntSlider(value=0, min=1, max=16, step=1)</pre>		
					295			
					296 297			
	E and the set							
	Funwave-tvd	~			298 299	<pre>run_items = [Box([Label(value='The number of nodes', layout = Layout(width = '350px')), numno</pre>		
					300	Box([Label(value= The number of hodes, layout = Layout(width = 350px)), humno Box([Label(value='The number of Processors of each node', layout=Layout(width =		
					301	layout= run item layout),		
					302	Box([runBtn]),		
	Innut	Run	Output	Visualization	303			
	Input	Run	Output	visualization	304			
					305			
					306	name_value_pairs = {		
					307	"PX" : str(numnodeSlider.value),		
	The number of node	s O		1	308	"PY" : str(numprocSlider.value)		
		0		•	309	}		
					310	with open("input funwave/input tmp.txt", "r") as tmp:		
	The number of Proce	accore of each node	\frown	4	311	with open ("input funwave/input.txt", "w") as inputfile:		
	The number of Floce	essors of each node		4	312	for line in tmp.readlines():		
					313	$g = re.match("^(PX PY) \s*=\s*(\s+)", line)$		
	and the second sec				314	if g:		
	Run				315	name = g.group(1)		
					316	if name in name_value_pairs:		
					317	<pre>inputfile.write(name+" = "+name_value_pairs[name]+"\n")</pre>		
					318	else:		
					319	inputfile.write(line)		
					320	inputfile.close()		
					321	<pre>tmp.close()</pre>		
					322			
						<pre>def runfun_btn_clicked(a):</pre>		
					324	<pre>if (modelTitle.value == "Funwave-tvd"):</pre>		
					325	modifyFWipput()		

- modifyFWinput() 326 cmd("rm -fr input")
- cmd("mkdir input") 327

System--output

In [2]: import model

Funwave-tvd	~				
Input	Run	Output	Visualization		
List jobs history					
List job output			Abort		
Download					

jobListBtn = Button(description='List jobs history', button style='primary', layc jobSelect = Select(layout = Layout(height = '150px', width='100%')) jobOutputBtn = Button(description='List job output', button_style='primary', layc abortBtn = Button(description='Abort', button_style='danger', layout= Layout(widt outputSelect = Select(layout = Layout(height = '150px', width='100%')) downloadOpBtn = Button(description='Download', button_style='primary', layout= La def jobList_btn_clicked(a): cout = cmd("jobs-list -1 10") out1 = cout["stdout"] jobSelect.options = out1 jobListBtn.on_click(jobList_btn_clicked) def abort btn clicked(a): g = re.match(r'^\S+',jobSelect.value) if q: jobid = g.group(0) rcmd = "jobs-stop "+jobid cmd(rcmd) abortBtn.on click(abort btn clicked) **def** jobOutput btn clicked(a): g = re.match(r'^\S+', jobSelect.value)

System--visualization

import model							
Funwave-tvd		~		475 fwYopt 476 fwplot 477 fwoneL 478	<pre>tion = Dropdown(options=['Choose one', 'eta', 'u', 'v']) sInter = widgets.interactive(fwOneD, Y_in_plots = fwYoption) Box = Box([fwplotsInter])</pre>		
Input	Bun	Output	Visualization	480 depthE 481 depthC 482 483	<pre>480 depthBtn = Button(description='display',button_style='primary', layou 481 depthOutput = widgets.Output() 482 483</pre>		
					<pre>484 485 def depth_Btn_clicked(a): 485 print (fw_para_pairs['Nglob']) 486 print (fw_para_pairs['Nglob']) 489 with depthOutput: 489 display(waterDepth(fw para pairs['Nglob'], fw para pairs['Nglob']) 489 display(waterDepth(fw para pairs['Nglob']) 489 display(waterDepth(fw para pairs['Nglob'], fw para pairs['Nglob']) 480 display(waterDepth(fw para pairs['Nglob'], fw para pairs['Nglob'], fw para pairs['Nglob']) 480 display(waterDepth(fw para pairs['Nglob']) 480 display(waterDepth(fw para pairs['Nglob'], fw para pairs['Nglob']) 480 display(waterDepth(fw para pairs['Nglob']) 480 display(waterDepth(fw para pairs['Nglob'], fw para pairs['Nglob']) 480 display(waterDepth(fw para pairs['Nglob']) 480 display(waterDepth(f</pre>		
÷1D				492 depthE 493 494	<pre>1 depthBtn.on_click(depth_Btn_clicked) 2 depthBox = Box([Label(value='Water Depth'),depthBtn],layout = Layout(v 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4</pre>		
Y_in_plots	Choose one	~		496 depPro 497 depPro 498 499	<pre>fileN = IntSlider(value=0, min=0, max=200) fileInter = widgets.interactive(depFofile, N = depFrofileN) fileBox = Box([Label(value='Depth Profile Snapshot'), depFrofile</pre>		
▶ 2D		501 depPro 502 503 depPro 504 depPro 505 def de	501 depProfileAnimaRange = FloatRangeSlider(value=[5,7], min=0.0, max=3 description='Time period (8):',ree 503 depProfileAnimBtn = Button(description='display',button_style='prin 504 depProfileAnimOutput = widgets.Output() 505 def depProfileAnim_Btn_clicked(a): 506 anim = depProfileWithEta(depProfileAnimaRange.value[0], depProfileAnimaRange.value[0], depProfileAnimaR				
► 3D							
	Input TD Y_in_plots (Funwave-tvd Input Run ✓1D Y_in_plots Choose one >2D	Funwave-tvd V Input Run Output • 1D	Funwave-tvd Input Run Output Visualization V_in_plots Choose one 2D	Funwave-tvd Input Run Output Visualization Imput Y_in_plots Choose one Y_in_plots Choose one Y_in_plots Imput Y_in_plots Choose one Y_in_plots Imput Y_in_plots Imput Y_in_plots Imput		

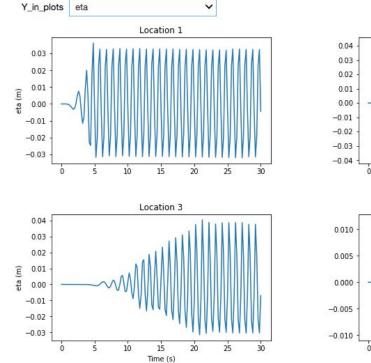
System--Future Work

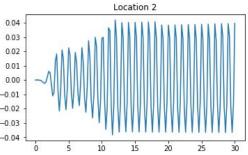
- Update OS, MPI, and source code for Funwave-tvd or Swan once a month, automatically (in progress).
- Intend to add more models

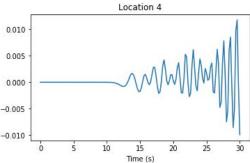


Results--1D

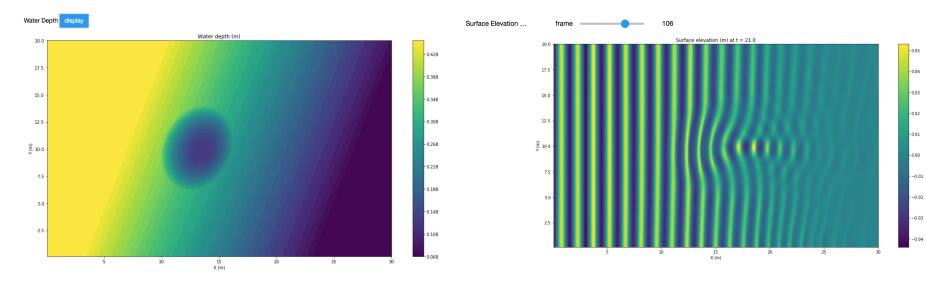
- 1D



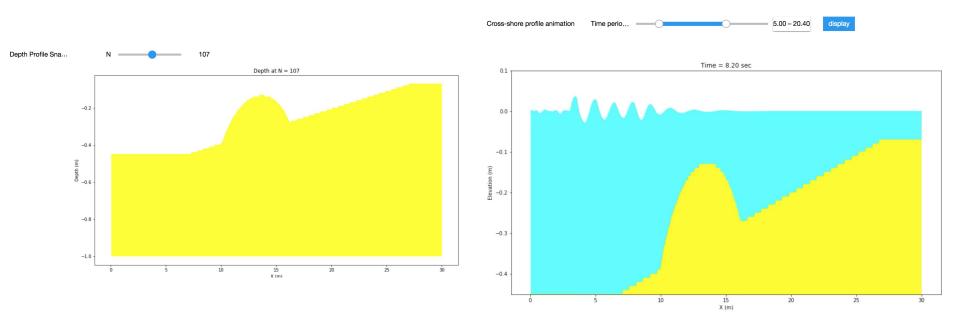




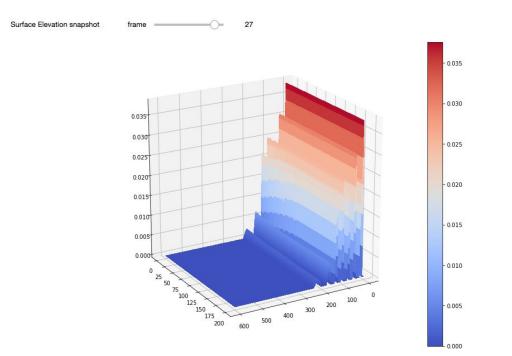
Results--vertical 2D











Conclusion

The CMR (Coastal Model Repository)

is targeting cloud and cloud-like architectures to enable quick deployment of coastal models and their working environment

serve as a community repository for precompiled open source models that are widely used by coastal researchers

Thanks!

