**Introduction**

Climate change can have adverse impacts on the strength of storms. Even modest changes in ocean surface temperature can have a significant impact on hurricane strength, making coastal regions increasingly vulnerable to storm surge.

- SLOSH (Seam, Lake, and Overland Surges from Hurricanes) is a computational model to predict storm surges in coastal areas.
- Scientists usually run a large ensemble of SLOSH instances to cope with errors and uncertainties in storm tracks and landfall location.

In this project, we develop generalized tools for rapid and cost-effective deployment of a large number (between 500 and 15,000) of small tasks on cloud resources. The project develops a pipeline framework for running ensemble simulations on the cloud.

We use the SLOSH model as the specific motivating application. Users who could benefit from the application include the National Hurricane Center who is a partner on the project, Federal Emergency Management Administration (FEMA), the U.S. Army Corps of Engineers, and state and local emergency managers.

![Image of Maximum Envelope of Water for a Hypothetical Storm of Category 1](image)

**SLOSH Execution Model**

The SLOSH instances generate a number of output files (for a number of groups) that record Maximum Envelope of Winds (MEOWs) and Maximum of Maximum of MEOWs (MOMs).

Dividing output files into groups facilitates interactive visualization and analysis; each group is captured by three parameters: storm direction, forward motion, and storm category.

**Job Scheduling in Cloud**

- Users submit jobs using a web portal. A Service Manager (e.g., Azure Daemon) fetches the submitted jobs and schedules them in the cloud resources. The Service Manager balances loads across worker nodes through partitioning the SLOSH instances.
- The worker nodes run the SLOSH instances and locally merge the output files generated. A separate merge process aggregates the intermediate file across the worker nodes.
- The Scheduler within the Service Manager reduces storage and I/O overheads in handling and aggregating intermediate output files.
- We use two approaches: a MapReduce runtime (Twister4Azure) and a Sigiri Middleware. Users are able to effect tradeoffs between cost and delay metrics.

![Figure: SLOSH ensemble execution model](image)

**Experiments**

<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>Average</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>141</td>
<td>2000</td>
<td>8000</td>
<td>5000</td>
</tr>
<tr>
<td>257</td>
<td>6000</td>
<td>7000</td>
<td>8000</td>
</tr>
<tr>
<td>348</td>
<td>7000</td>
<td>8000</td>
<td>8000</td>
</tr>
<tr>
<td>385</td>
<td>8000</td>
<td>8000</td>
<td>8000</td>
</tr>
</tbody>
</table>

**Execution time with varying track files (total VM = 20)**

![Figure: Scheduling tasks in Azure using Sigiri Middleware](image)

**Ongoing Efforts**

**Elastic processing:** Revise the leased resources in an on-line fashion depending submitted loads.

**Metadata harvest:** Automatic capture of metadata and provenance for the SLOSH output datasets to contribute towards trust and to reduce the burden of sharing the datasets. The metadata could be used to find which SLOSH simulation contributed each of the max values in the MEOWs/MOMs.

Develop a simple web-based interface (UI) for the system.

**References**