Novel Database and Usage Analytics for CESM Climate Model

First Steps to Tracking Worldwide Configuration and Performance

Lolita Mannik
Regis University
National Center for Atmospheric Research (NCAR)
Summer Internships in Parallel Computational Science (SIParCS)
Overview

1. Background on Community Earth System Model (CESM)
2. Model’s configuration
3. Data preparation and analysis
4. Key findings
5. Conclusion and future work

Image: http://www.cesm.ucar.edu/models/
Create a database dedicated to tracking broader CESM usage and performance

- Learn how the scientists use the model
- Track computational performance
CESM Climate Model

- Virtual laboratory
- Freely available
- Components:
  - Atmosphere
  - Land
  - Ocean
  - River
  - Sea and Land Ice
  - Wave

CESM = Community Earth System Model
Resolution

Coarse

Fine
Resolution

Coarse

Fine

+ the atmosphere!

Resolution

+ the atmosphere!
+ the ocean!

Coarse           Fine
**Method**

Data Engineering
- Acquiring
- Saving

---

```
TIMING PROFILE

Case          : b.e21.BHIST.f09_g17.CMIP6-historical.001
LID           : 2979765.chadmin1.181015-050236
User          : cmip6
Curr Date     : Mon Oct 15 10:01:22 2018
grid           :
compset        : HIST_CAM60_CLM50_BGC-CROP_CICE_POP2_ECO%ABIO-DIC_MOSART_CISM2%NOEVOLVE_WW3_BGC%BDRD
run_type       : hybrid, continue_run = TRUE (inittype = FALSE)
stop_option    : nyears, stop_n = 5
run_length     : 1825 days (1825.0 for ocean)
Init Time      : 63.817 seconds
Run Time       : 17837.627 seconds
Final Time     : 0.057 seconds
```
Method

1. Data Engineering
   - Acquiring
   - Saving

2. Data Wrangling
   - Reindexing
   - More parsing
   - Set data types
   - Intuitive columns
   - Calculations
Data Wrangling

Parse Run_Length

3650 days (3650.0 for ocean)

1. Strip everything after "days" in run_length column
   ```python
df['run_length_temp'] = df['run_length'].str.split('(^)').str[0]
   
   # Confirm every run_length contains the same units of days
   substr = 'days'
   print("Rows in df:", len(df))
   print("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
   
   Rows in df: 5160
   Rows with units of days: 5160
   ```

2. Strip "days" in run_length column
   ```python
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
   
   array(['3650', '365', '730', '2', '31', '1825', '2189', '1095', '5840', '5475', '1460', '2190', '5', '1', '10950', '7300', '4014', '426', '90', '4379'], dtype=object)
   ```

3. Convert necessary columns to numeric format
   ```python
   for col in df.columns:
       if 'length_days' in col:
           df[col] = pd.to_numeric(df[col])
   ```
Parse Run_Length

3650 days (3650.0 for ocean)

1. Strip everything after "days" in run_length column
   ```python
df['run_length_temp'] = df['run_length'].str.split('(').str[0]
```

2. Confirm every run_length contains the same units of days
   ```python
   substr = 'days'
   print("Rows in df:", len(df))
   print("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
   ```
   Rows in df: 5160
   Rows with units of days: 5160

3. Strip "days" in run_length column
   ```python
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
  ```
   ```
   array([3650, 365, 730, 2, 31, 1825, 2189, 1095, 5840, 5475, 1460, 2190, 5, 1, 10950, 7300, 4014, 426, 90, 4379], dtype=object)
   ```

4. Convert necessary columns to numeric format
   ```python
   for col in df.columns:
       if 'length_days' in col:
           df[col] = pd.to_numeric(df[col])
   ```
parse run_length

1. Strip everything after "days" in run_length column
   ```python
df['run_length_temp'] = df['run_length'].str.split('\(').str[0]
   ```

2. Strip "days" in run_length column
   ```python
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
   ```

3. Convert necessary columns to numeric format
   ```python
   for col in df.columns:
       if 'length_days' in col:
           df[col] = pd.to_numeric(df[col])
   ```
Data Wrangling

**Parse Run_Length**

#1. Strip everything after "days" in run_length column
```
# Confirm every run_length contains the same units of days
substr = 'days'
print("Rows in df:", len(df))
print("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
```

Rows in df: 5160
Rows with units of days: 5160

#2. Strip "days" in run_length column
```
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
```

```
array(['3650', '365', '730', '2', '31', '1825', '2189', '1095',
      '5840', '5475', '1460', '2190', '5', '1', '10950', '7300',
      '4014', '426', '90', '4379'], dtype='|O')
```

# Convert necessary columns to numeric format
```
for col in df.columns:
    if 'length_days' in col:
        df[col] = pd.to_numeric(df[col])
```
#1. Strip everything after "days" in run_length column
```
df['run_length_temp'] = df['run_length'].str.split('(').str[0]
```

#Confirm every run_length contains the same units of days
```
substr = 'days'
print("Rows in df:", len(df))
print("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
```

Rows in df: 5160
Rows with units of days: 5160

#2. Strip "days" in run_Lengt column
```
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
```

```
array(['3650 ', '365 ', '730 ', '2 ', '31 ', '1825 ', '2189 ', '1095 ',
      '5840 ', '5475 ', '1460 ', '2190 ', '5 ', '1 ', '10950 ', '7300 ',
      '4014 ', '426 ', '90 ', '4379 '], dtype=object)
```

#Convert necessary columns to numeric format
```
for col in df.columns:
    if 'length_days' in col:
        df[col] = pd.to_numeric(df[col])
```
#1. Strip everything after "days" in run_length column

```python
df['run_length_temp'] = df['run_length'].str.split('(').str[0]
```

# Confirm every run_length contains the same units of days

```python
substr = 'days'
print("Rows in df:", len(df))
print("Rows with units of days:", df.run_length_temp.str.count(substr).sum())
```

Rows in df: 5160
Rows with units of days: 5160

#2. Strip "days" in run_length column

```python
df['run_length_days'] = df['run_length_temp'].str.split('d').str[0]
df.run_length_days.unique()
```

```python
array(['3650', '365', '730', '2', '31', '1825', '2189', '1095', '5840', '5475', '1460', '2190', '5', '1', '10950', '7300', '4014', '426', '90', '4379'], dtype=object)
```

# Convert necessary columns to numeric format

```python
for col in df.columns:
    if 'length_days' in col:
        df[col] = pd.to_numeric(df[col])
```
# Data Wrangling

## Parse Run_Length

### #1. Strip everything after "days" in run_length column

```python
def['run_length_temp'] = def['run_length'].str.split(')').str[0]
```

# Confirm every run_length contains the same units of days
```python
substr = 'days'
print("Rows in df:", len(def))
print("Rows with units of days:", def.run_length_temp.str.count(substr).sum())
```

Rows in df: 5160
Rows with units of days: 5160

### #2. Strip "days" in run_length column

```python
def['run_length_days'] = def['run_length_temp'].str.split('d').str[0]
def.run_length_days.unique()
```

```
array(['3650', '365', '730', '2', '31', '1825', '2189', '1095', '5840', '5475', '1460', '2190', '5', '1', '10950', '7300', '4014', '426', '90', '4379'], dtype=object)
```

### #3. Convert necessary columns to numeric format

```python
for col in def.columns:
    if 'length_days' in col:
        def[col] = pd.to_numeric(def[col])
```
Method

1. Data Engineering
   - Acquiring
   - Saving

2. Data Wrangling
   - Reindexing
   - More parsing
   - Set data types
   - Intuitive columns
   - Calculations

3. Data Storage
Method

1. Data Engineering
   - Acquiring
   - Cleaning

   - Data types
   - Intuitive columns
   - Calculations

2. Data Storage

   SQL

   JSON

   USABLE DATA!!
Method

45 unique component configurations

7 unique grid configurations
Method

4. Exploratory Data Analysis
5. Statistical Analysis
Method

- 4. Exploratory Data Analysis
- 5. Statistical Analysis
- 6. Visualization
Analysis: Yearly Totals

361 Days

106 Unique Experiments

18,469 Simulated Years

118,824,082 CPU Hours
Power Equivalence

118,824,082 CPU Hours

or

189 trips around the equator in a Nissan Leaf

or

Annual power for 156 Colorado homes
Analysis: Grouping Atmospheric Configurations

Atmospheric Configuration vs. Model Cost

Model Cost
(CPU hrs / Simulated Year)
Analysis: Grouping Atmospheric Configurations

Atmospheric Configuration vs. Model Cost

Model Cost
(CPU hrs/Simulated Year)
Analysis: Grouping Atmospheric Configurations

Atmospheric Configuration vs. Model Cost

Model Cost (CPU hrs/Simulated Year)
Analysis: Grouping Atmospheric Configurations

Atmospheric Configuration vs. Model Cost

Model Cost (CPU hrs/Simulated Year)

Whole Atmosphere (WACCM)
Analysis: Grouping Atmospheric Configurations

Atmospheric Configuration (Grouped) vs. Model Cost

Model Cost (CPU hrs/Simulated Year)
Analysis: Atmospheric Components

CPU Hours

CAM vs. WACCM

<table>
<thead>
<tr>
<th>Month</th>
<th>CAM</th>
<th>WACCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aug-18</td>
<td>75%</td>
<td>25%</td>
</tr>
<tr>
<td>Sep-18</td>
<td>75%</td>
<td>25%</td>
</tr>
<tr>
<td>Oct-18</td>
<td>75%</td>
<td>25%</td>
</tr>
<tr>
<td>Nov-18</td>
<td>75%</td>
<td>25%</td>
</tr>
<tr>
<td>Dec-18</td>
<td>75%</td>
<td>25%</td>
</tr>
<tr>
<td>Jan-19</td>
<td>75%</td>
<td>25%</td>
</tr>
<tr>
<td>Feb-19</td>
<td>75%</td>
<td>25%</td>
</tr>
<tr>
<td>Mar-19</td>
<td>75%</td>
<td>25%</td>
</tr>
<tr>
<td>Apr-19</td>
<td>75%</td>
<td>25%</td>
</tr>
<tr>
<td>May-19</td>
<td>75%</td>
<td>25%</td>
</tr>
<tr>
<td>Jun-19</td>
<td>75%</td>
<td>25%</td>
</tr>
</tbody>
</table>
Analysis: System Upgrade

Simulations that span the early July upgrade

% Difference in Mean Model Cost

<table>
<thead>
<tr>
<th>Case ID</th>
<th>% Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
</tr>
</tbody>
</table>

# of CPUs
- 828
- 1728
- 3564
- 4320
Useful model of CESM database in the cloud
Conclusion

- Build worldwide CESM database
Future Work

Current data capture

Climate Modeling → Timing Files → Data Prep & Transform → Save to SQL → Analysis in Python
Future Work

Streamline data capture
Future Work

Streamline data capture

Feature engineering and machine learning
- Predict performance on configurations
- Verify correct installation and optimal settings for remote users
Acknowledgements

John Dennis
Brian Dobbins
Expert mentors

AJ Lauer
Virginia Do
Expert intern managers

Alice Bertini
SQL Training

References

Images
Unless otherwise noted, graphics are from www.vecteezy.com
Questions?

Lolita Mannik
lolita@FourWindsPublications.com

This presentation is posted at:
www.FourWindsPublications.com