CGEM Research Project

- Project: Start ~5/1/2018
- PI: Dr. John Lehrter  Marine Science
- Student: Lisa Lowe
- Code: CGEM
- Purpose: Evaluate Bioactivity in Estuaries and Coastal Ecosystems:
  - eutrophication*
  - hypoxia**
  - …

* Excessive richness of nutrients for plants from runoff.
** Deficiency in O2 (for animal life).
GCEM Research Project

- Problem:
  - Initial runs on Comet are taking way too long, 5x; cannot finish X-year runs.
  - Worked “Fine” on Naval Lab machine. No idea of what the problem is.
  - Naval system and Comet are similar; no access to Naval system.

- Problem with problem:
  - No Comparison--- Affinity, MPI, IO, Compiler Options
  - Requirement: phdf5 library.
GCEM Research Project

Finding 5x loss in performance should be trivial, right?

● stage I
  ✓ Verify: Get Input, Job Script, and Executable
  ✓ Use HPC environment
  ✓ Check Process layout
  ✓ Reduce Problem to a short run.

-- Run as HPC staff
-- htop for real-time, amask

● Stage II
  ✓ Check Scaling (for persistence)
  ✓ Investigate Code/Libraries causing problems
GCEM

- Check execution layout: 16/24 MPI tasks

Example on Stampede2: Binding to HW threads

```bash
export I_MPI_PIN_PROCESSOR_LIST="0-47"
```

NO CHANGE from default

```bash
$ htop
```
GCEM -- htop

- Check Process: Affinity mask.

```
$ htop
#scroll to a process, type a,
#scroll to [], hit space to toggle
```
GCEM -- amask

- Check Process
  
  ```
  $ export I_MPI_PIN_PROCESSOR_LIST="0-47"
  $ mpirun -np 48 amask_mpi
  #substitute amask_mpi for app
  ```

<table>
<thead>
<tr>
<th>rank</th>
<th>0000</th>
<th>0001</th>
<th>0002</th>
<th>0046</th>
<th>0047</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Core-ID</td>
<td>HWT 0 =</td>
<td>HWT 1 =</td>
<td>HWT 1 =</td>
<td>HWT 1 =</td>
<td>HWT 1 =</td>
</tr>
<tr>
<td>0000</td>
<td>0001</td>
<td>0002</td>
<td>0046</td>
<td>0047</td>
<td></td>
</tr>
</tbody>
</table>
Stage II
✓ Recompile
✓ Check Scaling
→ IO
Simulation
Data

Also, check on another system

IO Performance 4-hr Simulation, 16 tasks -- SDSC comet

\[ y = 61 + 20x \]

IO Performance 4-hr Simulation, 16 tasks -- TACC Skylake

\[ y = 24 + 1.4x \]

/oasis/projects/nsf/uot121
GCEM -- strace view in htop

Use *htop* again -- *strace* shows system calls *(select a process, hit s)*

---

What was learned:  
* MPI-IO, rank 0 performing IO  *(collected from other ranks)*
Important things coming to (or in) htop

2.1: Percent CPU/IO Delays (scheduling wait times)
2.1: IO READ/WRITE (Rates)
3.0: Hardware Performance Monitors
3.0: Multiple Screens

Available now, tree-view

https://hisham.hm/htop/
GCEM -- htop I/O Rates!

Go into setup mode F2, down to “Columns”, over to Available Columns, down to IO_WRITE_RATE, then F5
GCEM -- strace view through attachment

- In *htop* *strace* is limited – no timing information
- **cmd line ** *strace* has many options (attachment, timings, proc select, etc.)
  - Can attach to a process (-p *pid*) or groups -p "`/sbin/pidof a.out`"
  - Time consumed (-T)
  - Select|exclude Sys Calls (-e *trace*=one_or_more {memory, ipc, network, signal, file, csl of call names, etc.})

  e.g. `open("/dev/null", O_RDONLY) = 3  < 0.123456>`
GCEM -- strace output

$ strace -T -e trace=open -p 348107
strace: Process 348107 attached

open("./NETCDF/output.average.nc", O_RDONLY) = 31 <0.000491>
open("/dev/shm/Intel_MPI_6J1NS4", O_RDWR|O_CREAT|O_EXCL, 0600) = 31 <0.000022>
open("./NETCDF/output.average.nc", O_RDWR|O_CREAT, 0644) = 31 <0.003249>
open("/dev/shm/Intel_MPI_BejwC6", O_RDWR|O_CREAT|O_EXCL, 0600) = 32 <0.000021>
open("/work/00770/milfeld/.../data/2006/ssh.2006010207", O_RDWR) = 32 <0.000488>
open("/work/00770/milfeld/.../data/2006/t3d.2006010209", O_RDWR) = 32 <0.000388>
open("/work/00770/milfeld/.../data/2006/s3d.2006010209", O_RDWR) = 32 <0.000406>
open("/work/00770/milfeld/.../data/2006/zkh.2006010209", O_RDWR) = 32 <0.000768>
open("/work/00770/milfeld/.../data/2006/flx.2006010207", O_RDWR) = 32 <0.000568>
open("/work/00770/milfeld/.../data/2006/t3d.2006010206", O_RDWR) = 32 <0.000538>
open("/work/00770/milfeld/.../data/2006/s3d.2006010206", O_RDWR) = 32 <0.000503>
GCEM on skylake

```
$ strace -c -p 7010
strace: Process 7010 attached
^Cstrace: Process 7010 detached
```

<table>
<thead>
<tr>
<th>% time</th>
<th>seconds</th>
<th>usecs/call</th>
<th>calls</th>
<th>errors</th>
<th>syscall</th>
</tr>
</thead>
<tbody>
<tr>
<td>40.81</td>
<td>0.580647</td>
<td>690</td>
<td>842</td>
<td></td>
<td>write</td>
</tr>
<tr>
<td>28.81</td>
<td>0.409905</td>
<td>233</td>
<td>1762</td>
<td></td>
<td>read</td>
</tr>
<tr>
<td>24.50</td>
<td>0.348567</td>
<td>1</td>
<td>251629</td>
<td></td>
<td>sched_yield</td>
</tr>
<tr>
<td>4.49</td>
<td>0.063898</td>
<td>5</td>
<td>11778</td>
<td></td>
<td>process_vm_readv</td>
</tr>
<tr>
<td>0.44</td>
<td>0.006317</td>
<td>11</td>
<td>574</td>
<td></td>
<td>open</td>
</tr>
<tr>
<td>0.42</td>
<td>0.005949</td>
<td>10</td>
<td>574</td>
<td></td>
<td>close</td>
</tr>
<tr>
<td>0.22</td>
<td>0.003083</td>
<td>5</td>
<td>586</td>
<td></td>
<td>stat</td>
</tr>
<tr>
<td>0.16</td>
<td>0.002211</td>
<td>2</td>
<td>1194</td>
<td></td>
<td>lseek</td>
</tr>
<tr>
<td>0.09</td>
<td>0.001257</td>
<td>2</td>
<td>574</td>
<td></td>
<td>fstat</td>
</tr>
<tr>
<td>0.04</td>
<td>0.000551</td>
<td>1</td>
<td>574</td>
<td></td>
<td>ioctl</td>
</tr>
<tr>
<td>0.02</td>
<td>0.000258</td>
<td>29</td>
<td>9</td>
<td></td>
<td>fsync</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.00</td>
<td>1.422643</td>
<td>270096</td>
<td>574</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Discovery: *fsync* is the culprit

<table>
<thead>
<tr>
<th>TIME:</th>
<th>9.215647</th>
<th>Routine: fsync</th>
<th>TIME:</th>
<th>0.279867</th>
<th>Routine: fsync</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME:</td>
<td>8.583495</td>
<td>Routine: fsync</td>
<td>TIME:</td>
<td>0.000598</td>
<td>Routine: fsync</td>
</tr>
<tr>
<td>TIME:</td>
<td>9.825726</td>
<td>Routine: fsync</td>
<td>TIME:</td>
<td>0.083662</td>
<td>Routine: fsync</td>
</tr>
<tr>
<td>TIME:</td>
<td>13.452719</td>
<td>Routine: fsync</td>
<td>TIME:</td>
<td>0.000510</td>
<td>Routine: fsync</td>
</tr>
<tr>
<td>TIME:</td>
<td>13.263266</td>
<td>Routine: fsync</td>
<td>TIME:</td>
<td>0.006495</td>
<td>Routine: fsync</td>
</tr>
<tr>
<td>TIME:</td>
<td>13.846324</td>
<td>Routine: fsync</td>
<td>TIME:</td>
<td>0.080706</td>
<td>Routine: fsync</td>
</tr>
<tr>
<td>TIME:</td>
<td>1.653631</td>
<td>Routine: fsync</td>
<td>TIME:</td>
<td>0.000356</td>
<td>Routine: fsync</td>
</tr>
<tr>
<td>TIME:</td>
<td>3.530447</td>
<td>Routine: fsync</td>
<td>TIME:</td>
<td>0.009919</td>
<td>Routine: fsync</td>
</tr>
<tr>
<td>TIME:</td>
<td>10.318308</td>
<td>Routine: fsync</td>
<td>TIME:</td>
<td>0.005985</td>
<td>Routine: fsync</td>
</tr>
</tbody>
</table>
Beyond -- Modules

Changing your own library and application environments is a pain.

- Installing Applications, Tools and Libraries in user space:
  - Managing execution environments is taxing for ECSS staff
  - Asking user to manage an Application’s Environment, is “inconvenient”.
Beyond -- Better Living through mkmod

- **mkmod** -- creates and installs Modules in user-space
  - Installs modulefile for App|Tool|Lib -- just set **NAME, VER** and **TOPDIR** env variables
  - Includes setting **MODULEPATH** for $HOME/modulefiles in startups
  - sets prerequisites for the MPI and Compiler used for application
Beyond -- mkmod scenario

### Tool Installation

- $ cd $HOME
- $ git clone .../github.com/tacc/remora
- $ cd remora
- $ install.sh

...#vv setup info ...

set REMORA_BIN=$HOME/remora/bin
set PATH ...
set LD_LIBRARY_PATH ...

### Module Creation

- $ export NAME=remora
- $ export VER=1.8.2
- $ export TOPDIR=$HOME/remora
- $ export ENV1="REMORA_BIN=$HOME/remora/bin"
- $ mkmod

### Module Usage

- $ module load my_remora
- $#or
- $ module load my_remora/1.8.2  #specific version
Beyond -- Remora: Monitoring App Execution

- **REMORA** --Reports Resource Usage (for each node)
  - **Summaries**: Total CPU Time, Max Memory, MPI percentage, IO info
  - **Timelines** (download html, open master html page, click on monitored resource google plots)

```bash
$ module load my_remora
$ remora mpirun -np 24 my_mpi_app
$ tar -cvf remora.tar remora_*; scp remora.tar me@laptop:
```

```
laptop$ tar -xvf remora.tar
laptop$ cd remora_* #suffix is jobid
laptop$ open remora_summary.html
```
Beyond -- To Affinity and Beyond

- **amask** --Reports Affinity Mask for Each Parallel Process
  - Execute `amask_omp`, `amask_mpi` or `amask_hybrid` before executing parallel application
  - or Can execute same-named function in code (no arguments)
  - Optionally reports Kernel mask (proc-ids) or “Core mask”

```
$ cd
$ export I_MPI_DOMAIN_...
$ export OMP_PROC_BIND=spread OMP_PLACES=cores
$ module load amask

$ mpirun --np 24 amask_hybrid
$ mpirun --np 24 my_hybrid_app
```

<table>
<thead>
<tr>
<th>rank</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>proc-id</th>
</tr>
</thead>
</table>
| 0000 | 4
| 0001 | 5
| 0002 | 6
| 0003 | 7
| 0004 | 8
| 0005 | 9
| 0006 | 0
| 0007 | 1

This is for a pure MPI run (amask_mpi)
References

- https://hisham.hm/htop/
- https://github.com/tacc/remora
- https://github.com/tacc/mkmod
- https://github.com/tacc/amask