CICE Performance In CESM

Tony Craig
June 20, 2012
CESM SEWG Meeting
Motivation

• CICE performance is typically on the critical path with respect to overall CESM performance and in some cases is playing an important role in limiting CESM throughput
• With the current decompositions, CICE runs effectively on only limited pe counts

Goals

• Understand better where time is spent in CICE
• Understand better how CICE cost varies as we change the decomposition or increase the number of processes
• Explore alternative decompositions depending on the outcome of the above assessments
• Look for some low hanging fruit wrt code optimization
• Reduce the cost of CICE in CESM and provide more opportunities to run on a wider range of processes
Results:
Net CICE performance improvement on hopper

- Using updated decompositions and masked halos seems to
  - improve performance by > 20% for most cases
  - allow us to run on relatively arbitrary processor counts

at gx1v6 (time to run 20 days)
  16 pes, 272s → 212s (22%)
  64 pes, 91s → 69s (24%)
  320 pes, 27s → 21s (22%)
  1280 pes, 19s → 10s (43%)

at tx0.1v2 (time to run 10 days)
  1200 pes, 582s → 361s (38%)
  4800 pes, 148s → 112s (24%)
  18000 pes, 68s → 68s (0%)
CICE Performance in CESM1.1

• Computation
  – Physics – computations only where there is sea ice (therm1, therm2)
  – Radiation – new dEdd implementation is expensive. computations only where there is sea ice and the sun is up.
  – EVP subcycling – stress/stepu, cycled about 100 times per cice timestep
  – Horizontal Transport

• Communication (halo updates)
  – EVP subcycling, cycled about 100 times per cice timestep
  – Horizontal Transport (tracer updates)
  – bound_state (therm2, etc)

*Performance is ultimately limited by slowest processor*
Performance of CICE

- Computational performance driven by load imbalance to zeroth order, dominant term at low process counts and important at high process counts
  - CICE computations generally done only where there is sea ice
  - Large areas of the CICE grid never have any sea ice
  - The radiation computation done where there is sea ice and the sun is up
  - CICE varies seasonally and sun angle varies on diurnal cycle and seasonal timescales
- Communication is nearest neighbor halo update, it is a critical term at high process counts
  - Want to minimize number of messages and size of messages and maximize relative nearness of neighbors in communication network
Decomposition is a Critical Performance Tuning Knob

- CICE decomposes the horizontal grid across processes (tasks and/or threads)
- For a given global grid size, “nx” by “ny”
- Define a blocksize, “bx” x “by”
- Label the blocks
- Distribute blocks to processes using some decomposition strategy
  1. Cartesian Square-POP
  2. Cartesian SlenderX1
  3. Cartesian SlenderX2
  4. Spacecurve (Dennis)
  5. Roundrobin (New-ish)
  6. Blkrobin (New)
  7. Blkcart (New)
1. cartesian square-pop

Blocks aligned into contiguous “squares”
blocksize = 8 x 6, 8 blocks per processor

Good
• size of halo (low aspect ratio)
• number of neighbors (8)

OK
• limited pe counts

Poor
• cice load balance (N/S and radiation)
• land block elimination

block allocation to processors
2. cartesian slenderX1

- Blocks aligned “vertically”, span entire J index space
- blocksize = 1 x 48, 8 blocks/processor

Good
- cice load balance (N/S and radiation)
- number of neighbors (2)

OK
- size of halo (high aspect ratio)

Poor
- limited pe counts (320, 160, 80, 64, etc)
- land block elimination
3. cartesian slenderX2

Blocks aligned “vertically”, span half J index space
blocksize = 2 x 24, 8 blocks per processor

Good
- cice load balance (N/S)
- number of neighbors (5)

OK
- cice load balance (radiation)
- size of halo (high aspect ratio)

Poor
- limited pe counts (640, 320, 160, 80, 64, etc)
- land block elimination
4. spacecurve*

Blocks allocated linearly along spacecurve after land block elimination
blocksize = 8 x 6, 8 blocks/processor

Good
- size of halo
- number of neighbors (~8)
- flexible pe counts
- land block elimination

Poor
- cice load balance (N/S and radiation)

*credit to John Dennis
5. roundrobin

Blocks allocated round robin (left to right) after land block elimination
blocksize = 8 x 6, 8 blocks/processor

Good
- cice load balance (N/S and radiation)
- arbitrary pe counts
- land block elimination

Poor
- size of halo (local blocks not contiguous)
- number of neighbors (2+6*number of blocks/processor)

pes 80-90

block allocation to processors
6. blkrobin

Blocks allocated round robin “grouped” (back and forth) after land block elimination
blocksize = 8 x 6, 8 blocks/processor

Good
• cice load balance (N/S and radiation)
• arbitrary pe counts
• land block elimination

OK
• size of halo (local blocks partly contiguous)
• number of neighbors (26)

pes 80-90

block allocation to processors
7. blkcart

Blocks allocated into quadrants preserving neighbors.
blocksize = 8 x 6, 8 blocks/processor

Good
• cice load balance (N/S and radiation)
• number of neighbors (8)

OK
• size of halo (local blocks partly contiguous)
• somewhat flexible pe counts (multiples of 4 blocks per processor)

Poor
• land block elimination
### Grading CICE Decompositions

<table>
<thead>
<tr>
<th>decomposition</th>
<th>cice load balance north/south</th>
<th>cice load balance radiation</th>
<th>number of neighbors</th>
<th>amount of data to communicate</th>
<th>land block elimination in decomp</th>
<th>flexibility wrt pe counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. cartesian square-pop</td>
<td>F</td>
<td>F</td>
<td>B</td>
<td>A</td>
<td>F</td>
<td>C</td>
</tr>
<tr>
<td>2. cartesian slenderX1</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>3. cartesian slenderX2</td>
<td>B</td>
<td>C</td>
<td>B</td>
<td>B</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4. spacecurve</td>
<td>D</td>
<td>D</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>5. roundrobin</td>
<td>A</td>
<td>A</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>6. blkrobin</td>
<td>B</td>
<td>A</td>
<td>C</td>
<td>C</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>7. blkcart</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>F</td>
<td>C</td>
</tr>
</tbody>
</table>
CICE Timing Testing
for gx1v6, 320 pes, hopper
(time for 20 days)
CICE Code Changes

• New Decompositions
• Masked Halos
  – Most of the data “haloed” in CICE is unnecessary
  – Can update the halo data structure on the fly quickly to remove both messages and gridcells that don’t need to be communicated
  – There may be some overhead in setting up the masked halo
  – Set via namelist, default is “on”
• Overlapping Work and Communication
  – Attempted in subcycling with limited success
  – Works well if communication and work are about the same
  – Load imbalance across processes impacts effectiveness
  – Has some overhead
  – Set via namelist, default is “off”
Results:
Net CICE performance improvement on hopper

- Using updated decompositions and masked halos seems to
  - improve performance by > 20% for most cases
  - allow us to run on relatively arbitrary processor counts

at gx1v6 (time to run 20 days)
  16 pes, 272s -> 212s (roundrobin+masked halos, 20x48, 8)
  64 pes, 91s -> 69s (blkrobin+masked halos, 10x24, 8)
  320 pes, 27s -> 21s (slenderX2+masked halos, 2x96, 2)
  1280 pes, 19s -> 10s (spacecurve+masked halos, 8x6, 2)

at tx0.1v2 (time to run 10 days)
  1200 pes, 582s -> 361s (blkrobin+masked halos, 40x30, 6)
  4800 pes, 148s -> 112s (blkrobin+masked halos, 15x15, 8)
  18000 pes, 68s -> 68s (spacecurve, 6x6, 14)
Status

- bit-for-bit validation continues (displaced pole/tripole, threading, decomp, masked halos, various hardware)
- Running performance tests on other platforms
- Updating automatic cice decomp generation tool to provide “reasonable” default decomp for all resolutions and pe counts
- Improved weighting for spacecurve decomp being explored (John Dennis)
- Hope to have an updated CICE version in CESM1.1 in July
- High resolution exploration on Yellowstone high priority
Conclusions

• Using updated decomps and masked halos seems to
  – improve performance by > 20% for most cases
  – allow us to run on relatively arbitrary processor counts
• Now have a better sense of how the CICE performance varies with resolution and decomposition – how do we share this information with the community?
• Determining the optimal block size, decomposition, and thread count for a given resolution, target processor count, and hardware still requires testing
• Still want to understand performance better at highest resolution and highest processor counts
• Future Work?
  – “CICE performance simulator”
  – Other decomposition strategies
  – Allow distinct dynamics and physics decompositions (like CAM)
  – Persistent Communication (Monika Lücke, GRS)
  – More detailed algorithm profiling to identify poorly performing kernels