Advances in parallel-split dynamics and physics

Aaron S. Donahue and Peter M. Caldwell
Lawrence Livermore National Laboratory
(ACME Software Modernization CMDV project)
ACME All-Hands Breakout #4
June 6, 2017

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 IM Release Number LLNL-PRES-XXXXXX
For intellectual and numerical tractability, climate models are broken into components:

- The coarsest granularity is “dynamics” (fluid flow) and “physics” (diabatic processes)
- These two processes must be brought together by a loose coupling mechanism
- The three most common are:
  - Sequential-tendency-splitting (STS), $se_{ftype}=0$
  - Sequential-update-splitting (SUS), $se_{ftype}=1$
  - Parallel-splitting (PS), $se_{ftype}=3$ (proposed)
Process Splitting: Coupling Strategies

Sequential-Tendency Split (STS, aka no step splitting): The tendency from Proc1 is used by Proc2

Sequential-Update Split (SUS, aka time split/fractional steps): State is updated after each process

Parallel Split (PS, aka process/additive split): All processes are computed from the same state
Process Splitting: Domain Decomposition
(A) The Earth is divided into a cubed sphere of quadrilateral elements.

(B) Dynamics is solved on individual spectral elements.

\[(1^\circ) = 5.4K\text{ elements}\]
\[
\left(\frac{1}{4}\right)^\circ = 86.4K\text{ elements}
\]
Figure 1: Dynamics and physics domains for the ACME model. (A) cubed sphere, (B) example spectral element, (C) example physics column. Image credit: Dennis et al. (2012) Int. J. of High Performance Computing Applications (A and B) and Neale et al. (2010) CAM 4.0 (C)
**Process Splitting: Current Scalability**

Sequential-Tendency Split (STS, aka no step splitting):
- Limited by smallest domain,
  - Max Cores = \# elements

Parallel Split (PS, aka process/additive split):
- Potential to scale up to largest domain,
  - Max Cores = \# columns

---

**Performance Computing Applications.**

<table>
<thead>
<tr>
<th>Dynamics</th>
<th>Physics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (1)^0 ) = 5.4K elements, 48.6K columns</td>
<td></td>
</tr>
<tr>
<td>( (1/4)^0 ) = 86.4K elements, 777.6K columns</td>
<td></td>
</tr>
</tbody>
</table>

---

Process Splitting: Current Scalability

Sequential-Tendency Split (STS, aka no step splitting):

Parallel Split (PS, aka process/additive split):

Fig: Fraction of ACME v1alpha7 integration time spent in various processes.
Parallel-Split: Implementation

1. Bugfix to allow for dynamics to be solved on a subset of total atmosphere model cores. Namelist variable `dyn_npes` now works on master. (PR #1393)

2. Adjust `phys_grid` subroutine to only assign columns “chunks” to physics solving cores.
Parallel-Split: Potential Issues

1. Bugfix to allow for dynamics to be solved on a subset of total atmosphere model cores. Namelist variable `dyn_npes` now works on master. (PR #1393)

2. Adjust `phys_grid` subroutine to only assign columns “chunks” to physics solving cores.

1. Is there a degraded solution due to change in dynamics/physics coupling mechanism?

2. How do we handle the mass conservation violations inherent in using a parallel-split approach?

3. How to implement within current code infrastructure?

4. Do we actually accomplish improved performance?
Parallel-Split: degraded solution? Solution looks good

- We have implemented the parallel splitting technique in ACME v0 for dyn and phys, using the same computational cores for both processes, on a 1° domain.
- 10 years of simulation have shown that the method is stable, provided that $\Delta t = 900s$.
- Comparison with sequential-tendency splitting (default) shows good results!

**Fig:** precipitation rate from 10 year ACME v0 runs with; parallel-state splitting (top), sequential-tendency splitting (bottom) and difference (right).
Parallel-Split: mass conservation? New approaches are promising

Possible to have fluxes that remove more mass than is available.

Leading to negative mass in an element.

A. **Clipping**: Setting all negative masses to zero.

B. **Weighted Horizontal Distribution**: Drawing mass from neighboring nodes horizontally.

C. **Weighted Vertical Distribution**: Drawing mass from neighboring levels vertically.

D. **Full Element Distribution**: Drawing mass from all points within an element.
Parallel-Split: mass conservation? New approaches are promising

Possible to have fluxes that remove more mass than is available.

Leading to negative mass in an element.

A. **Clipping**: Setting all negative masses to zero.

B. **Weighted Horizontal Distribution**: Drawing mass from neighboring nodes horizontally.

C. **Weighted Vertical Distribution**: Drawing mass from neighboring levels vertically.

D. **Full Element Distribution**: Drawing mass from all points within an element.

**Fig**: Daily average mass conservation corrections

Clipping increases the liquid cloud water mass by ~ 0.7% per timestep.

Weighted distribution methods dramatically improve the mass conservation properties.
Parallel-Split: implement? Yes, almost...

Limit at # of elements, fixed for standard model in PR #1393.

Kinks in performance evidence of poor physics column load balancing.

Standard approach flattens out early on, while parallel-split continues to improve, outperforming the standard approach at maximum core count.

We are still having coding issues with producing output.

Figure 6: Solution timings for 10-day simulations with no output on the 7.5 degree (ne4) mesh for the standard model and for parallel-split implementation.
Parallel-Split: improved performance? Not yet...

We see a degradation of the model scalability for core counts less than the number of elements for the parallel-split implementation.

We do not see improved performance for core counts greater than the number of elements, STS still outperforms PS except at the max core count option or a few “optimum” cases.

Figure 6: Solution timings for 10-day simulations with no output on the 7.5 degree (ne4) mesh for the standard model and for parallel-split implementation.
Parallel-Split: improved performance? Solution ideas:

- A 50/50 split of dynamics and physics cores is inefficient and leads to dynamics cores sitting idle for long periods.
- Improved performance in terms of solving physics and dynamics separately is traded for increased communication costs.

Figure 7: Average computational cost per core for dynamics (dynamics), physics (physics AC and BC) and dynamics-physics communication (d_p and p_d coupling) for the standard ACME model (left) and the parallel-split implementation (right). Top panels represent only the cores assigned to dynamics, bottom panels are cores assigned to physics.
Parallel-Split: improved performance? Solution ideas:

- A 50/50 split of dynamics and physics cores is inefficient and leads to **dynamics cores sitting idle** for long periods.
- Improved performance in terms of solving physics and dynamics separately is traded for **increased communication costs**.

- Determine and implement **optimum** balancing of dynamics and physics computational cores.
- Implement a more sophisticated distribution of cores assigned to dynamics and physics such that most dynamics/physics communication is **inter-compute-node**.

---

**Figure 7:** Average computational cost per core for dynamics (dynamics), physics (physics AC and BC) and dynamics-physics communication (d_p and p_d coupling) for the standard ACME model (left) and the parallel-split implementation (right). Top panels represent only the cores assigned to dynamics, bottom panels are cores assigned to physics.
Parallel-Split: improved performance? Optimum balancing:
Steps forward and other applications:

• Fix issues with output in parallel-split implementation.
• Implement a more balanced distribution of dynamics and physics cores over a single computational node.
• Improvement and further testing of mass conservation techniques.

• Possible implementation of parallel-split approach using the product of the next-gen coupler project (also a part of the CMDV project).
• Application of parallel-split in with the current work being conducted on super-parameterization.
For more info come check out my poster: A13

Thank you!