Subcolumns in CAM

- Cheryl Craig and Steve Goldhaber
Outline

The CAM subcolumn infrastructure provides the ability to create fields with several elements within a single grid column.

- Rationale for subcolumns
- An introduction to infrastructure changes made to support subcolumns
- Examples of subcolumn usage in CAM physics
- How subcolumns are created and managed
Subcolumns in CAM

- Rationale for subcolumns
- An introduction to infrastructure changes made to support subcolumns
- Examples of subcolumn usage in CAM physics
- How subcolumns are created and managed
Rationale for subcolumns

- Allow parameterizations finer granularity than traditional grid box
- Ability to run ensemble parameterizations within a single column
- Don’t need to increase resolution on entire model to study one parameterization in detail

(Satellite view of earth)
Standardizing subcolumns

- Subcolumns already exist in CAM – radiation, SPCAM branch, etc.
- Each implementation is specialized – “shoehorned” into old existing static structures
- Not able to share subcolumns between parameterizations due to unique implementations
Subcolumns in CAM

• Rationale for subcolumns
• An introduction to infrastructure changes made to support subcolumns
• Examples of subcolumn usage in CAM physics
• How subcolumns are created and managed
Features of changes

- `state%ncol` is still the number of columns to loop over inside parameterizations
  - underlying physics parameterization do not require code modifications - work the same whether grid or subcolumns
- Variable number of subcolumns per grid column
- Grid and/or subcolumn fields only allocated as requested
- `state` (for grid) and `state_sc` (for subcolumns) may both exist at same time – user responsible for keeping them synchronized
- Several subcolumn generators may exist in CAM, but only one will be used per run
Dynamic state/tend/ptend changes

- Variables unchanged within new subcolumn framework
  - `pcols` - maximum number of grid columns
  - `state%ncol` - number of columns to loop over inside parameterizations
    - may be larger than `pcols` is using subcolumns
- New implementation to support subcolumns
  - `psubcols` - maximum number of subcolumns = 1 for grid
  - `state%psetcols` - maximum number of total columns
    - whether using grid or subcolumns = `pcols*psubcols`
    - replaces `pcols` in a lot of places
- `state/tend/ptend` are now dynamically allocated
New Fields to support subcolumns

Subcolumnized data

Conceptual Layout

<table>
<thead>
<tr>
<th>Grid columns</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subcolumns</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

X = Data locations

Internal Storage Layout - compressed

<table>
<thead>
<tr>
<th>Grid Columns</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subcolumns

pcols = 6  ngrdcol = 5
psubcols = 4  nsubcol(6) = (2,1,3,1,1,0)
psetcols = 24  ncol = 8
indcol(24) = (1,1,2,3,3,3,4,5,0,...)

ppgrid_parameter  state_variables

Grid data

<table>
<thead>
<tr>
<th>Grid columns</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subcolumns</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

pcols = 6  ngrdcol = 5
psubcols = 1  nsubcol(6) = (1,1,1,1,1,0)
psetcols = 6  ncol = 5
indcol(6) = (1,2,3,4,5,0)
Physics buffer (pbuf) changes

pbuf structure contains:

(buffer_field_type):: bfg%data – holds grid data
(buffer_field_type):: bfg_sc%data – holds subcolumn data – NEW

- bfg%data and bfg_sc%data are only allocated as requested
- Each physics buffer field can have grid-only, subcolumn-only or both grid and subcolumn data
New control parameters

**col_type:** int  
- 0=grid, 1=subcolumns  
Used to identify WHICH field (used in `pbuf_get_field` for example)

**grid_type:** int(bit_field_kind)  
- each bit is turned on/off to indicate which field(s) are required

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>...</th>
<th>Grid and subcolumn</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>Grid only</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>Subcolumn only</td>
</tr>
</tbody>
</table>

Used to identify ALL fields which are currently turned on (used in `pbuf_add_field` for example)
Subcolumns in CAM

- Rationale for subcolumns
- An introduction to infrastructure changes made to support subcolumns
- Examples of subcolumn usage in CAM physics
- How subcolumns are created and managed