Coupling Workshop Overview

May 27, 2014

> There are only two hard things in computer science: cache invalidation, naming things and off-by-one errors.
>
> – *Phil Karlton*

General remarks

- cell variables are now accessed through methods (so called *accessors*), e.g. `cells[cellId].m_isInterface` becomes `a_isInterface(cellId)`
- all accessors have an `a_` prefix to make them visually distinct from regular class/instance methods
- inside the blocks, some *grid* accessors are only available through the `grid()` method: `childIds`, `noChildIds`, `parentIds`, `globalIds`, `levels`, `weights`, `workLoads`, `noOffsprings` (e.g. `grid().a_childId(...)`)
- all other accessors are directly available in the blocks and grids, e.g. `a_coordinate(...)`
- all accessors accept the `cellId` as the first argument (`gCellId` in case of levelset variables)
- all variable names are translated as close as possible to their accessor names
  - generally, the singular form is used for all accessors, e.g. `m_childIds` becomes `a_childId(...)`
  - accessors that return the number of elements of something use the plural form, e.g. `noChildIds` becomes `a_noChildren(...)`
  - `m_noNghbrIds` is now `a_hasNeighbor(...)` since this is what the variable is used for
- all cell variables are made `private` and suffixed with an underscore (e.g. `m_childIds_`) to prevent accidental usage
- underscores are also appended to local variables that shadow one of the accessors (e.g. `ZFSId parentId = 0;` becomes `ZFSId parentId_ = 0;`)
- except for the removal of some pointers-to-cell, no changes or improvements to the code logic are performed

How to access cell variables

- in order to obtain a value, use the accessor return value, e.g.

  ```
  ZFSId pId = a_parentId(cellId);
  ```

- the accessor can also be assigned to, thus changing a value is as easy as e.g.

  ```
  a_coordinate(cellId, dir) = 0.0;
  ```

- never use the cell variables directly anymore, always use the accessors instead
- access to `a_hasProperty` (replaces `b_properties`) must be done using the `Cell::<propname>` syntax (using integers does not work anymore)

Detailed changes

All large-scale changes were done using the scripts in `aux/regex/`. The scripts for each rename were put in a folder with the name of the renamed variable. There is also a script available to automatically append underscores to variables that would otherwise shadow an accessor.

Changes in the grid generators

Variable name (*old*)	Accessor name (*new*)	Contact
cells[cellId].b_properties[p]	a_hasProperty(cellId, Cell<prop>)	Michael
cells[cellId].m_childIds[pos]	a_childId(cellId, pos)	Jerry
cells[cellId].m_coordinates[dir]	a_coordinate(cellId, dir)	Michael
cells[cellId].m_delete	a_isToDelete(cellId)	Hsun-Jen
cells[cellId].m_globalId	a_globalId(cellId)	Hsun-Jen
cells[cellId].m_isBndryCell	a_isBndryCell(cellId)	Vitali
cells[cellId].m_isBndryGhostCell	a_isBndryGhostCell(cellId)	Vitali
cells[cellId].m_isInterface	a_isInterface(cellId)	Vitali
cells[cellId].m_level[0]	a_level(cellId)	Michael
cells[cellId].m_nghbrIds[dir][0]	a_neighborId(cellId, dir)	Jerry
cells[cellId].m_noChildIds	a_noChildren(cellId)	Michael
cells[cellId].m_noNghbrIds[dir]	a_hasNeighbor(cellId, dir)	Hsun-Jen

Variable name (*old*)	Accessor name (*new*)	Contact
cells[cellId].m_noOffsprings	a_noOffsprings(cellId)	Vitali
cells[cellId].m_oldVariables[varId]	a_oldVariable(cellId, varId)	Hsun-Jen
cells[cellId].m_parentId	a_parentId(cellId)	Gonzalo
cells[cellId].m_periodicDirection	a_periodicDirection(cellId)	Vitali
cells[cellId].m_quasiPeriodicDirection	a_quasiPeriodicDirection(cellId)	Vitali
cells[cellId].m_rcnstrctnNghbrIds[n]	a_reconstructionNeighborId(cellId, n)	Michael
cells[cellId].m_variables[varId]	a_variable(cellId, varId)	Michael
cells[cellId].m_weight	a_weight(cellId)	Hsun-Jen
cells[cellId].m_workLoad	a_workLoad(cellId)	Vitali
m_pCells[cellId].m_rfnDistance	a_refinementDistance(cellId)	Vitali

Changes in all blocks

Variable name (*old*)	Accessor name (*new*)	Contact
cells[cellId].b_properties[p]	a_hasProperty(cellId, Cell::<prop>)	Michael
cells[cellId].m_childIds[pos]	a_grid().childId(cellId, pos)	Jerry
cells[cellId].m_coordinates[dir]	a_coordinate(cellId, dir)	Michael
cells[cellId].m_globalId	a_grid().globalId(cellId)	Michael
cells[cellId].m_level[0]	a_grid().level(cellId)	Michael
cells[cellId].m_nghbrIds[dir][0]	a_neighborId(cellId, dir)	Jerry
cells[cellId].m_noChildIds	a_grid().noChildren(cellId)	Michael
cells[cellId].m_noNghbrIds[dir]	a_hasNeighbor(cellId, dir)	Hsun-Jen
cells[cellId].m_noOffsprings	a_grid().noOffsprings(cellId)	Vitali
cells[cellId].m_oldVariables[varId]	a_oldVariable(cellId, varId)	Hsun-Jen
cells[cellId].m_parentId	a_grid().parentId(cellId)	Gonzalo
cells[cellId].m_variables[varId]	a_variable(cellId, varId)	Michael
cells[cellId].m_weight	a_grid().weight(cellId)	Hsun-Jen
cells[cellId].m_workLoad	a_grid().workLoad(cellId)	Vitali

Changes in FV block

Variable name (*old*)	Accessor name (*new*)	Contact
cells[cellId].m_bndryId	a_boundaryId(cellId)	Hsun-Jen
cells[cellId].m_delete	a_isToDelete(cellId)	Hsun-Jen
cells[cellId].m_dt1Variables[varId]	a_dt1Variable(cellId, varId)	Hsun-Jen
cells[cellId].m_dt2Variables[varId]	a_dt2Variable(cellId, varId)	Hsun-Jen
cells[cellId].m_isBndryCell	a_isBndryCell(cellId)	Vitali
cells[cellId].m_isBndryGhostCell	a_isBndryGhostCell(cellId)	Vitali
cells[cellId].m_isInterface	a_isInterface(cellId)	Vitali
cells[cellId].m_multilevelVars[dir]	a_multilevelVar(cellId, dir)	Vitali
cells[cellId].m_noRcnstrctnNghbrIds	a_noReconstructionNeighbors(cellId)	Hsun-Jen
cells[cellId].m_onlyBoundary	a_onlyBoundary(cellId)	Vitali
cells[cellId].m_periodicDirection	a_periodicDirection(cellId)	Vitali
cells[cellId].m_quasiPeriodicDirection	a_quasiPeriodicDirection(cellId)	Vitali
cells[cellId].m_rcnstrctnNghbrIds[n]	a_reconstructionNeighborId(cellId, n)	Michael
cells[cellId].m_reactionRate[0]	a_reactionRate(cellId)	Hsun-Jen
cells[cellId].m_restrictedRHS[varId]	a_restrictedRHS(cellId, varId)	Vitali
cells[cellId].m_slope[varId][dir]	a_slope(cellId, varId, dir)	Hsun-Jen
cells[cellId].m_spongeBndryId1	a_spongeBndryId(cellId, id=0)	Hsun-Jen
cells[cellId].m_spongeBndryId2	a_spongeBndryId(cellId, id=1)	Hsun-Jen
cells[cellId].m_spongeBndryId3	a_spongeBndryId(cellId, id=2)	Hsun-Jen
cells[cellId].m_spongeFactor	a_spongeFactor(cellId)	Hsun-Jen
cells[cellId].m_spongeFactorStart	a_spongeFactorStart(cellId)	Hsun-Jen
cells[cellId].m_tau[varId]	a_tau(cellId, varId)	Vitali
m_variables[cellId][varId]	a_pvariable(cellId, varId)	Michael

Changes in FV levelset block

All accessors for the levelset variables are suffixed with the letter `G` in order to distinguish them from normal cell variable accessors. Exceptions are made for the three accessors `parentGId`, `childGId`, and `neighborGId`, since they are already distinct.

Variable name (*old*)	Accessor name (*new*)	Contact
gCells[cellId].m_bandNghbrIds[n]	a_bandNeighborIdG(cellId, n)	Vitali
gCells[cellId].m_bodyId[set]	a_bodyIdG(cellId, set)	Vitali
gCells[cellId].m_childIds[pos]	a_childGId(cellId, pos)	Jerry
gCells[cellId].m_coordinates[dir]	a_coordinateG(cellId, dir)	Michael
gCells[cellId].m_curvature[set]	a_curvatureG(cellId, set)	Vitali
gCells[cellId].m_fExt[n]	a_fExtG(cellId, n)	Vitali
gCells[cellId].m_flameSpeed[n]	a_flameSpeedG(cellId, n)	Vitali
gCells[cellId].m_flowCellId	a_flowCellIdG(cellId)	Vitali
gCells[cellId].m_haloLayer	a_haloLayerG(cellId)	Vitali
gCells[cellId].m_hasBeenRefined	a_hasBeenRefinedG(cellId)	Vitali
gCells[cellId].m_inBand[set]	a_inBandG(cellId, set)	Hsun-Jen
gCells[cellId].m_inGamma[set]	a_inGammaG(cellId, set)	Hsun-Jen
gCells[cellId].m_inShadowLayer	a_inShadowLayerG(cellId)	Hsun-Jen
gCells[cellId].m_isBndryCell	a_isBndryCellG(cellId)	Vitali
gCells[cellId].m_isGBndryCell[set]	a_isGBoundaryCellG(cellId, set)	Hsun-Jen
gCells[cellId].m_isGHaloCell	a_isGHaloCellG(cellId)	Vitali
gCells[cellId].m_isGWindowCell	a_isGWindowCellG(cellId)	Vitali
gCells[cellId].m_levelSetFunction[set]	a_levelSetFunctionG(cellId, set)	Vitali
gCells[cellId].m_level[0]	a_levelG(cellId)	Michael
gCells[cellId].m_nghbrIds[dir][0]	a_neighborGId(cellId, dir)	Jerry
gCells[cellId].m_noChildIds	a_noChildrenG(cellId)	Michael
gCells[cellId].m_noNghbrIds[dir]	a_hasNeighborG(cellId, dir)	Hsun-Jen
gCells[cellId].m_noOffsprings	a_noOffspringsG(cellId)	Vitali
gCells[cellId].m_normalExtensionVelocity[n]	a_normalExtensionVelocityG(cellId, n)	Vitali
gCells[cellId].m_normalVector[n]	a_normalVectorG(cellId, n)	Vitali
gCells[cellId].m_parentId	a_parentGId(cellId)	Gonzalo
gCells[cellId].m_regridTrigger	a_regridTriggerG(cellId)	Hsun-Jen
gCells[cellId].m_stretch[set]	a_stretchG(cellId, set)	Vitali
gCells[cellId].m_weight	a_weightG(cellId)	Hsun-Jen

Changes in LBM block

Variable name (*old*)	Accessor name (*new*)	Contact
cells[cellId].m_bndId	a_bndId(cellId)	Vitali
cells[cellId].m_distributionsThermal[v]	a_distributionThermal(cellId, v)	Vitali
cells[cellId].m_distributions[v]	a_distribution(cellId, v)	Michael
cells[cellId].m_isBndryCell	a_isBndryCell(cellId)	Vitali
cells[cellId].m_isBndryGhostCell	a_isBndryGhostCell(cellId)	Vitali
cells[cellId].m_isInterface	a_isInterface(cellId)	Vitali
cells[cellId].m_isInterfaceChild	a_isInterfaceChild(cellId)	Vitali
cells[cellId].m_isInterfaceParent	a_isInterfaceParent(cellId)	Vitali
cells[cellId].m_kappa	a_kappa(cellId)	Vitali
cells[cellId].m_nu	a_nu(cellId)	Vitali
cells[cellId].m_oldDistributionsThermal[v]	a_oldDistributionThermal(cellId, v)	Vitali
cells[cellId].m_oldDistributions[v]	a_oldDistribution(cellId, v)	Michael
cells[cellId].m_onlyBoundary	a_onlyBoundary(cellId)	Vitali
cells[cellId].m_spongeFactor	a_spongeFactor(cellId)	Hsun-Jen

Changes in AVG block

Variable name (*old*)	Accessor name (*new*)	Contact
cells[cellId].m_fluctuations[v]	a_fluctuation(cellId, v)	Vitali
cells[cellId].m_vortexCriterion	a_vortexCriterion(cellId)	Vitali
cells[cellId].m_vorticity[dir]	a_vorticity(cellId, dir)	Vitali

Contact persons

Name	Phone
Gonzalo Brito Gadeschi	94821
Hsun-Jen Cheng	97025
Jerry Grimmen	95433

Name	Phone
Michael Schlottke	95188
Vitali Pauz	95188