PUGHReduce

Gabrielle Allen, Thomas Radke

Date: 2003/04/15 15:31:12

Abstract

Reductions operations which are performed using the PUGH driver

1 Purpose

This thorn registers a number of reduction operators with the flesh. The reductions are performed using internals of the PUGH driver, so that this thorn can only be used when `CactusPUGH/PUGH` is active.

The reduction operations this thorn registers are

Reduction Operator	Calculates	By
`average`*, `mean`*	the average/mean of a grid variable	$\sum GV/N$
`count`	the number of grid points in a grid variable	N
`maximum`*	the maximum of a grid variable	$\max GV$
`minimum`*	the minimum of a grid variable	$\min GV$
`norm1`, `L1Norm`	the L1 norm of a grid variable	$(\Sigma\|GV\|)/N$
`norm2`, `L2Norm`	the L2 norm of a grid variable	$\sqrt[2]{(\Sigma\|GV\|^2)/N}$
`norm3`, `L3Norm`	the L3 norm of a grid variable	$\sqrt[3]{(\Sigma\|GV\|^3)/N}$
`norm4`, `L4Norm`	the L4 norm of a grid variable	$\sqrt[4]{(\Sigma\|GV\|^4)/N}$
`norm_inf`, `LinfNorm`	the Infinitity norm of a grid variable	$\max\|GV\|$
`sum`*	the sum of the elements of a grid variable	$\sum GV$

Reduction operators with multiple names are just synonyms for the same kind of reduction operation. In the formulas GV is the grid variable to be reduced, and N denotes the number of its elements. Reduction operators marked with * cannot be applied to grid variables of complex datatype.

2 Examples

The following C example illustrates how the get the maximum value of a grid function.

```
int vindex;             /* grid variable index */
CCTK_REAL result;       /* resulting reduction value */
int target_proc;        /* processor to hold the result */
int reduction_handle;   /* handle for reduction operator */
char *reduction_name;   /* reduction operator to use */


/* want to get the maximum for the wavetoy grid function */
reduction_name = "maximum";
vindex = CCTK_VarIndex ("wavetoy::phi");

/* the reduction result will be obtained by processor 0 only */
target_proc = 0;
```

```
/* get the handle for the given reduction operator */
reduction_handle = CCTK_ReductionHandle (reduction_name);
if (reduction_handle >= 0)
{
  /* now do the reduction using the flesh's generic reduction API
    (passing in one input, expecting one output value of REAL type) */
  if (CCTK_Reduce (cctkGH, target_proc, reduction_handle,
                   1, CCTK_VARIABLE_REAL, &result, 1, vindex) == 0)
  {
    if (CCTK_MyProc (cctkGH) == target_proc)
    {
      printf ("%s reduction value is %f\n", reduction_name, result);
    }
  }
  else
  {
    CCTK_VWarn (1, __LINE__, __FILE__, CCTK_THORNSTRING,
                "%s reduction failed", reduction_name);
  }
}
else
{
  CCTK_VWarn (1, __LINE__, __FILE__, CCTK_THORNSTRING,
              "Invalid reduction operator '%s'", reduction_name);
}
```