



**Hewlett Packard**  
Enterprise

# Preparing an application for Hybrid Supercomputing



Larry Kaplan  
Senior Distinguished Technologist

John M Levesque  
Chief Technologist for Cray Programming Environment  
Senior Distinguished Technologist

✉ [John.levesque@hpc.com](mailto:John.levesque@hpc.com)



# Agenda

**Introduction and a Little History**

**Steps in moving an application to a GPU**

**A simple example - himeno**

**Optimizing himeno – with openmp on the node**

**Optimizing himeno – with openmp offload on the GPUs (or Openacc). DEMO**

**Dealing with Halo exchanges DEMO**

# Performance Analysis Tools with porting and tuning of applications on HPE systems

## Highlights:

- Different tools to fit different developer needs—from quick visual analysis to variety of different experiments, integration with compilers and more...
- Target **scalability** issues in all areas of tool development—designed to improve performance on the largest of systems
- Provide **whole program performance analysis** across many nodes to identify critical performance bottlenecks in a program
- Help to uncover issues but also **suggestions to improve performance**
- Unique and valuable **load imbalance analysis**
- Target **ease of use** with simple and advanced user interfaces



Our performance tools profiled production applications with over 256,000 ranks.

# Code Parallelization Assistant

The image displays three overlapping screenshots of the 'Reveal' code parallelization assistant interface.

- Top Left Screenshot:** Shows the 'Navigation' pane on the left with a tree view of loops. The 'Source' pane on the right displays C code for a loop starting at line 51. A status bar at the bottom indicates: 'Info - Line 51: A loop starting at line 51 was not vectorized because it contains a call to...'.
- Top Right Screenshot:** Shows the 'Scoping Results' pane for 'sweepz190: Loop@51'. It lists variables with their types and scopes, along with diagnostic messages. For example, 'f' is an 'Array' with 'Unresolved' status and a 'FAIL' message: 'Last defining iteration not known for variable that is live on exit.'.
- Bottom Left Screenshot:** Shows the 'OpenMP Directive' pane, which displays the generated OpenMP code for the selected loop, including directives like `$OMP parallel do default(none)` and `$OMP private (i,j,k,m,n,delp2,delp1,shock,temp2,old_flat,onemf,hd,t & ...)`.

- Reduce effort associated with adding OpenMP to MPI programs
- Works in conjunction with our compiler and performance tools
- Easily navigate through source code to highlighted dependences or bottlenecks
- Identify work-intensive loops to parallelize, perform dependence analysis, scope variables and generate OpenMP directives
- Great first step when moving large, complex loops to GPUs

# Directive-Based Programming Models

- Huge potential to provide cross-architecture portability (CPUs and GPUs)
- Standard specifications that all compiler vendors can implement
- Performance portability across vendors has been a recent challenge
- Has been critical for Fortran, especially for offloading
- OpenMP and OpenACC
  - Continued participation in language committees
  - Ongoing support for current and future specifications
  - Leverage common compiler and library codebase for OpenMP and OpenACC implementations
  - Significant opportunities for general performance improvements
  - Significant opportunities for improving construct-to-hardware mapping
    - Better cross-vendor consistency
    - Better use of descriptive features (e.g., “omp loop”)
    - Use of multidimensional grids, especially for “collapse” loops
- Specialized, low-level directive-based models may expand user base
  - Optimize performance for a limited set of OpenMP/OpenACC constructs and APIs
  - Provide a portable model similar to existing kernel languages (e.g., CUDA or HIP)

# Using perftools-lite ( or -loops or -hbm)

- module load perftools-lite or perftools-lite-loops or perftools-lite-hbm
  - Module perftools-base should already be loaded
- Build application
- Run application
- Statistics report comes out within standard out
  - Also generates a directory of profile data to be examined with different options



# Lets start with a simple example

- `cd himeno_hybrid`
- `module load perftools-lite`
- `ftn -rm himenoBMTxpr.f -ohimeno`

```
[levesque@o185i082 himeno_hybrid]$ ftn -rm himenoBMTxpr.f -ohimeno  
WARNING: PerfTools is saving object files from a temporary directory into directory  
'/lvol/levesque/.craypat/himeno/507153'  
INFO: creating the PerfTools-instrumented executable 'himeno' (lite-samples) ...OK
```

- `srun -n 2 ./himeno`



# Perftools-lite profile – Run on 1 nodes–2 MPI tasks/node

Table 1: Profile by Function

pat\_report -T himenox+507859-4242835s>profile\_T

| Samp%  | Samp    | Imb.<br>Samp | Imb.<br>Samp% | Group<br>Function<br>PE=HIDE |
|--------|---------|--------------|---------------|------------------------------|
| 100.0% | 6,421.0 | --           | --            | Total                        |
| 99.0%  | 6,359.0 | --           | --            | USER                         |
| 98.8%  | 6,341.0 | 51.0         | 1.6%          | jacobi_                      |
| 0.3%   | 18.0    | 0.0          | 0.0%          | initmt_                      |
| =====  |         |              |               |                              |
| 0.7%   | 47.5    | --           | --            | MPI                          |
| -----  |         |              |               |                              |
| 0.7%   | 46.5    | 45.5         | 98.9%         | MPI_WAITALL                  |
| 0.0%   | 0.5     | 0.5          | 100.0%        | MPI_ISEND                    |
| 0.0%   | 0.5     | 0.5          | 100.0%        | mpi_isend_                   |
| =====  |         |              |               |                              |
| 0.2%   | 14.5    | 4.5          | 47.4%         | ETC                          |
| -----  |         |              |               |                              |
| 0.2%   | 14.5    | 4.5          | 47.4%         | __cray_memcpy_ROME           |
| =====  |         |              |               |                              |

Exclusive time  
Sampling is in  
100<sup>th</sup> of a second

Imbalance

# Perftools-lite profile – Run on 1 nodes–2 MPI tasks/node

Table 2: Profile of maximum function times

pat\_report -T himenox+507859-4242835s>profile\_T

| Samp%  | Samp    | Imb. | Imb.  | Function           |
|--------|---------|------|-------|--------------------|
|        |         | Samp | Samp% | PE                 |
| 100.0% | 6,392.0 | 51.0 | 1.6%  | jacobi_            |
| 100.0% | 6,392.0 | --   | --    | pe.0               |
| 98.4%  | 6,290.0 | --   | --    | pe.1               |
| 1.4%   | 92.0    | 45.5 | 98.9% | MPI_WAITALL        |
| 1.4%   | 92.0    | --   | --    | pe.1               |
| 0.0%   | 1.0     | --   | --    | pe.0               |
| 0.3%   | 19.0    | 4.5  | 47.4% | __cray_memcpy_ROME |
| 0.3%   | 19.0    | --   | --    | pe.1               |
| 0.2%   | 10.0    | --   | --    | pe.0               |
| 0.3%   | 18.0    | 0.0  | 0.0%  | initmt_            |
| 0.3%   | 18.0    | --   | --    | pe.0               |
| 0.3%   | 18.0    | --   | --    | pe.1               |

Exclusive time  
Sampling is in  
100<sup>th</sup> of a second

Imbalance

# Perftools-lite profile – Run on 1 nodes–2 MPI tasks/node

Table 3: Profile by Group, Function, and Line

pat\_report -T himenox+507859-4242835s>profile\_T

| Samp%  | Samp    | Imb.<br>Samp | Imb.<br>Samp% | Group<br>Function<br>Source<br>Line<br>PE=HIDE         |
|--------|---------|--------------|---------------|--------------------------------------------------------|
| 100.0% | 6,421.0 | --           | --            | Total                                                  |
| 99.0%  | 6,359.0 | --           | --            | USER                                                   |
| 98.8%  | 6,341.0 | --           | --            | jacobi_<br>/lv01/levesque/himeno_hybrid/himenoBMTxpr.f |
| 4      | 1.6%    | 105.5        | 1.5           | 2.8%   line.89                                         |
| 4      | 0.0%    | 1.0          | 1.0           | 100.0%   line.213                                      |
| 4      | 85.9%   | 5,518.5      | 6.5           | 0.2%   line.214                                        |
| 4      | 5.5%    | 352.0        | 18.0          | 9.7%   line.224                                        |
| 4      | 1.6%    | 101.0        | 8.0           | 14.7%   line.225                                       |
| 4      | 3.5%    | 226.0        | 17.0          | 14.0%   line.226                                       |
| 4      | 0.6%    | 37.0         | 1.0           | 5.3%   line.227                                        |

# This is why we used -rm on ftn line (-h list=a)

```
200.      C*****
201.      subroutine jacobi(nn,gosa)
202.      C*****
203.      IMPLICIT REAL*4(a-h,o-z)
204.      C
205.      include 'mpif.h'
206.      include 'param.h'
207.      C
208.      + 1-----<      DO loop=1,nn
209.      1                gosa=0.0
210.      1                wgosa=0.0
211.      + 1 2-----<      DO K=2,kmax-1
212.      + 1 2 3-----<      DO J=2,jmax-1
213.      1 2 3 Vr3-----<      DO I=2,imax-1
214.      1 2 3 Vr3                S0=a(I,J,K,1)*p(I+1,J,K)+a(I,J,K,2)*p(I,J+1,K)
215.      1 2 3 Vr3                +a(I,J,K,3)*p(I,J,K+1)
216.      1 2 3 Vr3                1                +b(I,J,K,1)*(p(I+1,J+1,K)-p(I+1,J-1,K)
217.      1 2 3 Vr3                2                -p(I-1,J+1,K)+p(I-1,J-1,K))
218.      1 2 3 Vr3                3                +b(I,J,K,2)*(p(I,J+1,K+1)-p(I,J-1,K+1)
219.      1 2 3 Vr3                4                -p(I,J+1,K-1)+p(I,J-1,K-1))
220.      1 2 3 Vr3                5                +b(I,J,K,3)*(p(I+1,J,K+1)-p(I-1,J,K+1)
221.      1 2 3 Vr3                6                -p(I+1,J,K-1)+p(I-1,J,K-1))
222.      1 2 3 Vr3                7                +c(I,J,K,1)*p(I-1,J,K)+c(I,J,K,2)*p(I,J-1,K)
223.      1 2 3 Vr3                8                +c(I,J,K,3)*p(I,J,K-1)+wrk1(I,J,K)
224.      1 2 3 Vr3                9                SS=(S0*a(I,J,K,4)-p(I,J,K))*bnd(I,J,K)
225.      1 2 3 Vr3                WGOSA=WGOSA+SS*SS
226.      1 2 3 Vr3                wrk2(I,J,K)=p(I,J,K)+OMEGA *SS
227.      1 2 3 Vr3----->      enddo
228.      1 2 3----->      enddo
229.      1 2----->      enddo
230.      1                C
231.      + 1 2-----<      DO K=2,kmax-1
232.      + 1 2 3-----<      DO J=2,jmax-1
233.      1 2 3-----<      DO I=2,imax-1
234.      1 2 3-----<      p(I,J,K)=wrk2(I,J,K)
235.      1 2 3----->      enddo
236.      1 2 3----->      enddo
237.      1 2----->      enddo
```

# Perftools-lite profile – Run on 1 nodes–2 MPI tasks/node

Table 3: Profile by Group, Function, and Line

pat\_report -T himenox+507859-4242835s>profile\_T

| Samp%  | Samp    | Imb.<br>Samp | Imb.<br>Samp% | Group<br>Function<br>Source<br>Line<br>PE=HIDE         |          |
|--------|---------|--------------|---------------|--------------------------------------------------------|----------|
| 100.0% | 6,421.0 | --           | --            | Total                                                  |          |
| -----  |         |              |               |                                                        |          |
| 99.0%  | 6,359.0 | --           | --            | USER                                                   |          |
| -----  |         |              |               |                                                        |          |
| 98.8%  | 6,341.0 | --           | --            | jacobi_<br>/lv01/levesque/himeno_hybrid/himenoBMTxpr.f |          |
| -----  |         |              |               |                                                        |          |
| 4      | 1.6%    | 105.5        | 1.5           | 2.8%                                                   | line.89  |
| 4      | 0.0%    | 1.0          | 1.0           | 100.0%                                                 | line.213 |
| 4      | 85.9%   | 5,518.5      | 6.5           | 0.2%                                                   | line.214 |
| 4      | 5.5%    | 352.0        | 18.0          | 9.7%                                                   | line.224 |
| 4      | 1.6%    | 101.0        | 8.0           | 14.7%                                                  | line.225 |
| 4      | 3.5%    | 226.0        | 17.0          | 14.0%                                                  | line.226 |
| 4      | 0.6%    | 37.0         | 1.0           | 5.3%                                                   | line.227 |
| =====  |         |              |               |                                                        |          |

# Lets Optimize this code

- Shared memory parallelism
- Put on the GPU
- To do these things we need to understand the loop limits

```
module swap perftools-lite perftools-lite-loops
```

```
ftn -rm himenoBMTxpr.f -ohimnox
```

```
WARNING: PerfTools is saving object files from a temporary directory into  
directory '/lvol/levesque/.craypat/himnox/508047'
```

```
INFO: creating the PerfTools-instrumented executable 'himnox' (lite-  
loops) ...OK
```

```
srun -n 2 ./himnox
```



pat\_report -T -Oct himenox+508124-  
4242835t>profile\_Tloops\_ct

# Perftools-lite-loops profile – Run on 1 nodes–2 MPI tasks/node

| Table 1: Calltree with Loop Inclusive Time |              |           |               |                        |       |
|--------------------------------------------|--------------|-----------|---------------|------------------------|-------|
| Incl<br>Time%                              | Incl<br>Time | Loop Exec | Loop<br>Trips | Calltree<br>PE=HIDE    |       |
|                                            |              |           | Avg           |                        | Total |
| 100.0%                                     | 63.50        | --        | --            |                        |       |
| 100.0%                                     | 63.50        | --        | --            | himenobmtxp_           |       |
| 197.3%                                     | 125.31       | 2,331,148 | 126.5         | jacobi_                |       |
| 99.4%                                      | 63.14        | 2         | 74.0          | jacobi_.LOOP.1.li.208  |       |
| 98.6%                                      | 62.61        | 148       | 125.0         | jacobi_.LOOP.2.li.211  |       |
| 0.3%                                       | 0.21         | 18,500    | 125.0         | jacobi_.LOOP.6.li.232  |       |
| 0.2%                                       | 0.10         | 2,312,500 | 126.5         | jacobi_.LOOP.7.li.233  |       |
| 0.6%                                       | 0.35         | 16,256    | 128.5         | initmt_                |       |
| 0.3%                                       | 0.20         | 1         | 128.0         | initmt_.LOOP.1.li.154  |       |
| 0.3%                                       | 0.20         | 128       | 128.0         | initmt_.LOOP.2.li.155  |       |
| 0.3%                                       | 0.19         | 16,384    | 128.0         | initmt_.LOOP.3.li.156  |       |
| 0.2%                                       | 0.16         | 1         | 127.0         | initmt_.LOOP.4.li.175  |       |
| 0.0%                                       | 0.00         | 0         | --            | _STOP3                 |       |
| 0.0%                                       | 0.00         | --        | --            | initcomm_              |       |
| 0.0%                                       | 0.00         | 0         | --            | mpi_init_(sync)        |       |
| 0.0%                                       | 0.00         | 0         | --            | MPI_INIT               |       |
| 0.0%                                       | 0.00         | 0         | --            | mpi_finalize_(sync)    |       |
| 0.0%                                       | 0.00         | --        | --            | initmax_               |       |
| 0.0%                                       | 0.00         | 1         | 2.0           | initmax_.LOOP.1.li.352 |       |
| 0.0%                                       | 0.00         | 1         | 2.0           | initmax_.LOOP.2.li.359 |       |

# This is why we used -rm on ftn line (-h list=a)

```
200.      C*****
201.      subroutine jacobi(nn,gosa)
202.      C*****
203.      IMPLICIT REAL*4(a-h,o-z)
204.      C
205.      include 'mpif.h'
206.      include 'param.h'
207.      C
208.      + 1-----<      DO loop=1,nn
209.      1                gosa=0.0
210.      1                wgosa=0.0
211.      + 1 2-----<      DO K=2,kmax-1
212.      + 1 2 3-----<      DO J=2,jmax-1
213.      1 2 3 Vr3----<      DO I=2,imax-1
214.      1 2 3 Vr3                S0=a(I,J,K,1)*p(I+1,J,K)+a(I,J,K,2)*p(I,J+1,K)
215.      1 2 3 Vr3                +a(I,J,K,3)*p(I,J,K+1)
216.      1 2 3 Vr3                1                +b(I,J,K,1)*(p(I+1,J+1,K)-p(I+1,J-1,K)
217.      1 2 3 Vr3                2                -p(I-1,J+1,K)+p(I-1,J-1,K))
218.      1 2 3 Vr3                3                +b(I,J,K,2)*(p(I,J+1,K+1)-p(I,J-1,K+1)
219.      1 2 3 Vr3                4                -p(I,J+1,K-1)+p(I,J-1,K-1))
220.      1 2 3 Vr3                5                +b(I,J,K,3)*(p(I+1,J,K+1)-p(I-1,J,K+1)
221.      1 2 3 Vr3                6                -p(I+1,J,K-1)+p(I-1,J,K-1))
222.      1 2 3 Vr3                7                +c(I,J,K,1)*p(I-1,J,K)+c(I,J,K,2)*p(I,J-1,K)
223.      1 2 3 Vr3                8                +c(I,J,K,3)*p(I,J,K-1)+wrk1(I,J,K)
224.      1 2 3 Vr3                9                SS=(S0*a(I,J,K,4)-p(I,J,K))*bnd(I,J,K)
225.      1 2 3 Vr3                WGOSA=WGOSA+SS*SS
226.      1 2 3 Vr3                wrk2(I,J,K)=p(I,J,K)+OMEGA *SS
227.      1 2 3 Vr3---->      enddo
228.      1 2 3----->      enddo
229.      1 2----->      enddo
230.      1                C
231.      + 1 2-----<      DO K=2,kmax-1
232.      + 1 2 3-----<      DO J=2,jmax-1
233.      1 2 3-----<      DO I=2,imax-1
234.      1 2 3-----<      p(I,J,K)=wrk2(I,J,K)
235.      1 2 3----->      enddo
236.      1 2 3----->      enddo
237.      1 2----->      enddo
```

# Lets Optimize this code

- Shared memory parallelism
- We have a tool called reveal to help parallelize code for host and CPU
- We need to build a program library to use reveal

module unload perftools-lite-loops

ftn -hpl=himeno.pl himenoBMTxpr.f

reveal himeno.pl himenox+508124-4242835t



Navigation

◀ Nested Loop View

| Loop Name                | Inclusive Time (%) | Inclusive Time (secs) | Loops Hit |
|--------------------------|--------------------|-----------------------|-----------|
| ▼ jacobi().LOOP.1.li.208 | 99.40              | 63.14                 |           |
| jacobi().LOOP.2.li.211   | 98.60              | 62.61                 | 1         |

Source - /lvol/levesque/himeno\_hybrid/himenoBMTxpr.f

+ cce/14.0.1 Up Down Save

|      |     |                                                        |
|------|-----|--------------------------------------------------------|
| L    | 208 | D0 loop=1,nn                                           |
|      | 209 | gosa=0.0                                               |
|      | 210 | wgosa=0.0                                              |
| F    | 211 | D0 K=2,kmax-1                                          |
| F    | 212 | D0 J=2,jmax-1                                          |
| FVr3 | 213 | D0 I=2,imax-1                                          |
|      | 214 | S0=a(I,J,K,1)*p(I+1,J,K)+a(I,J,K,2)*p(I,J+1,K          |
|      | 215 | +a(I,J,K,3)*p(I,J,K+1)                                 |
|      | 216 | 1            +b(I,J,K,1)*(p(I+1,J+1,K)-p(I+1,J-1,K)    |
|      | 217 | 2            -p(I-1,J+1,K)+p(I-1,J-1,K))               |
|      | 218 | 3            +b(I,J,K,2)*(p(I,J+1,K+1)-p(I,J-1,K+1)    |
|      | 219 | 4            -p(I,J+1,K-1)+p(I,J-1,K-1))               |
|      | 220 | 5            +b(I,J,K,3)*(p(I+1,J,K+1)-p(I-1,J,K+1)    |
|      | 221 | 6            -p(I+1,J,K-1)+p(I-1,J,K-1))               |
|      | 222 | 7            +c(I,J,K,1)*p(I-1,J,K)+c(I,J,K,2)*p(I,J-1 |
|      | 223 | 8            +c(I,J,K,3)*p(I,J,K-1)+wrk1(I,J,K)        |
|      | 224 | 9            SS=(S0*a(I,J,K,4)-p(I,J,K))*bnd(I,J,K)    |
|      | 225 | WGOSA=WGOSA+SS*SS                                      |
|      | 226 | wrk2(I,J,K)=p(I,J,K)+OMEGA *SS                         |
|      | 227 | enddo                                                  |
|      | 228 | enddo                                                  |
|      | 229 | enddo                                                  |
|      | 230 | C                                                      |
| F    | 231 | D0 K=2,kmax-1                                          |
| F    | 232 | D0 J=2,jmax-1                                          |
| AF   | 233 | D0 I=2,imax-1                                          |
|      | 234 | p(I,J,K)+wrk2(I,J,K)                                   |

Info - Line 208

- "mpi\_allreduce" was not inlined because the compiler was unable to locate the routine.
- "sendp"(/lvol/levesque/himeno\_hybrid/himenoBMTxpr.f:446) not flattened because "mpi\_wait" was not inlined.
- "sendp\_clone 508738 1657189072 3"(/lvol/levesque/himeno\_hybrid/himenoBMTxpr.f:446) was not inlined.

himenox+508124-4242835t loaded.

File Edit View Help

Navigation

Nested Loop View

| Loop Name                | Inclusive Time (%) | Inclusive Time (secs) | Loops Hit |
|--------------------------|--------------------|-----------------------|-----------|
| ▼ jacobi().LOOP.1.li.208 | 99.40              | 63.14                 | 1         |
| ▼ jacobi().LOOP.2.li.211 | 98.60              | 62.61                 | 1         |

Source - /lvol/levesque/himeno\_hybrid/himenoBMTxpr.f

+ cce/14.0.1 Up Down Save

L 208 D0 loop=1,nn

209 gosa=0.0

210 wgosa=0.0

F 211 D0 K=2,kmax-1

F 212 D0 J=2,jmax-1

himenox+508124-4242835t loaded.

Scope Loops

Edit List Scoping Options

List of Loops to be Scoped

| Scope? | Line #                                  | File or Source Line                         |
|--------|-----------------------------------------|---------------------------------------------|
| ▼      | <input checked="" type="checkbox"/>     | /lvol/levesque/himeno_hybrid/himenoBMTxpr.f |
|        | <input checked="" type="checkbox"/> 211 | D0 K=2,kmax-1                               |

Apply Filter

Time: 0.000

Trips: 2

Threads: 4

Speedup: 0.010

Scope For CPU

Scope For GPU

Cancel

1 Loop selected

Close

himenox+508124-4242835t loaded.

A loop starting at line 212 is flat (contains no external calls)

**Source** - /lvol/levesque/himeno hybrid/himenoBMTxpr.f

### ◀ Nested Loop View

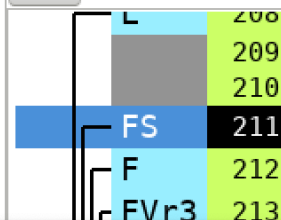
cce/14.0.1

Up

Down

Save

| Loop Name                                                                                                | Inclusive Time (%) | Inclusive Time (secs) | H |
|----------------------------------------------------------------------------------------------------------|--------------------|-----------------------|---|
| ▼ jacobi().LOOP.1.li.208                                                                                 | 99.40              | 63.14                 |   |
| jacobi().LOOP.2.li.211  | 98.60              | 62.61                 |   |



```

DO loop=1,nn
  gosa=0.0
  wgosa=0.0
DO K=2,kmax-1
  DO J=2,jmax-1
    DO I=2,imax-1

```

## Scope Loops

## Scoping Results

☒ **Reveal OpenMP Scoping**

### himenobMTxpr.f: Loop@211

| Name ▾         | Type   | Scope          | Info |
|----------------|--------|----------------|------|
| jmax           | Scalar | P <b>S</b> C U |      |
| k              | Scalar | <b>P</b> S C U |      |
| kmax           | Scalar | P <b>S</b> C U |      |
| omega          | Scalar | P <b>S</b> C U |      |
| p              | Array  | P <b>S</b> C U |      |
| s0             | Scalar | <b>P</b> S C U |      |
| ss             | Scalar | <b>P</b> S C U |      |
| wgosa <b>R</b> | Scalar | P <b>S</b> C U |      |
| wrk1           | Array  | P <b>S</b> C U |      |
| wrk2           | Array  | P <b>S</b> C U |      |

First/Last Private

☐ Enable FirstPrivate☐ Enable LastPrivate

### Reduction

None

Find Name:

Insert Directive

Show Directive

Close

● A loop starting at line 211 was not vectorized because a better candidate was found at line 2

**Source** - /lvol/levesque/himeno hybrid/himenoBMTxpr.f

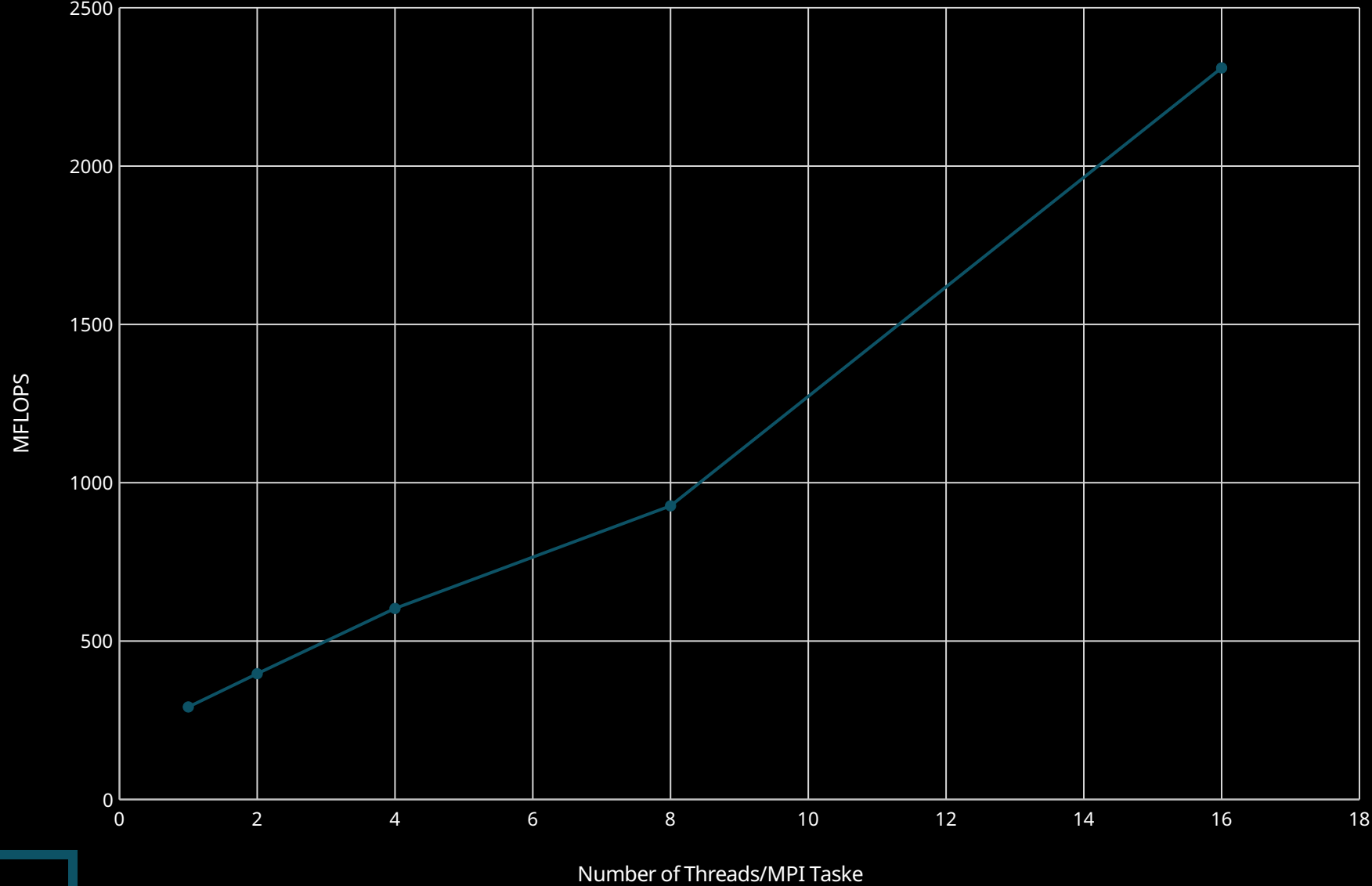
+



|      |     |                                                                       |
|------|-----|-----------------------------------------------------------------------|
|      | 205 | include 'mpit.h'                                                      |
|      | 206 | include 'param.h'                                                     |
|      | 207 | C                                                                     |
| L    | 208 | D0 loop=1,nn                                                          |
|      | 209 | gosa=0.0                                                              |
|      | 210 | wgosa=0.0                                                             |
|      |     | ! Directive inserted by Cray Reveal. May be incomplete.               |
|      |     | !\$OMP parallel do default(none)                                      |
|      |     | !\$OMP& shared(a, b, bnd, c, imax, jmax, kmax, omega, p, wgosa, wrk2) |
|      |     | !\$OMP& wrk2)                                                         |
|      |     | !\$OMP& private(i, j, k, s0, ss)                                      |
|      |     | !\$OMP& reduction(+: wgosa)                                           |
| FS   | 211 | D0 K=2,kmax-1                                                         |
| F    | 212 | D0 J=2,jmax-1                                                         |
| FVr3 | 213 | D0 I=2,imax-1                                                         |
|      | 214 | S0=a(I,J,K,1)*p(I+1,J,K)+a(I,J,K,2)*p(I,J+1,K)                        |
|      | 215 | 1 +a(I,J,K,3)*p(I,J,K+1)                                              |
|      | 216 | 2 +b(I,J,K,1)*(p(I+1,J+1,K)-p(I+1,J-1,K))                             |
|      | 217 | 3 -p(I-1,J+1,K)+p(I-1,J-1,K))                                         |
|      | 218 | 4 +b(I,J,K,2)*(p(I,J+1,K+1)-p(I,J-1,K+1))                             |
|      | 219 | 5 -p(I,J+1,K-1)+p(I,J-1,K-1))                                         |
|      | 220 | 6 +b(I,J,K,3)*(p(I+1,J,K+1)-p(I-1,J,K+1))                             |
|      | 221 | 7 -p(I+1,J,K-1)+p(I-1,J,K-1))                                         |
|      | 222 | 8 +c(I,J,K,1)*p(I-1,J,K)+c(I,J,K,2)*p(I,J-1,K)                        |
|      | 223 | 9 +c(I,J,K,3)*p(I,J,K-1)+wrk1(I,J,K)                                  |
|      | 224 | SS=(S0*a(I,J,K,4)-p(I,J,K))*bnd(I,J,K)                                |
|      | 225 | WGOSA=WGOSA+SS*SS                                                     |
|      | 226 | wrk2(I,J,K)=p(I,J,K)+OMEGA*SS                                         |

- "mpi\_allreduce" was not inlined because the compiler was unable to locate the routine.
- "sendp"(/lvol/levesque/himeno\_hybrid/himenoBMTxpr.f:446) not flattened because "mpi\_wait" was not inlined.
- "sendp\_clone\_508738\_1657189072\_3"(/lvol/levesque/himeno\_hybrid/himenoBMTxpr.f:446) was not flattened because "mpi\_wait" was not inlined.

Performance



# Now lets put it on the GPU

- Go back to Reveal and generate GPU code



Navigation

Loop Performance

63.1397 JACOBI@208

62.6147 JACOBI@211 (gpu)

0.2107 JACOBI@232

0.1961 INITMT@154

0.1960 INITMT@155

0.1871 INITMT@156

0.1576 INITMT@175

0.1039 JACOBI@233

Source - /lvol/levesque/himeno\_hybrid/himenoBMTxpr.f

208

209

210

211

DO loop=1,nn

gosa=0.0

wgosa=0.0

DO K=2,kmax-1

cce/14.0.1 Up Down Save

Reveal OpenMP Scoping

Scope Loops

Scoping Results

himenoBMTxpr.f: Loop@211 (gpu)

| Name | Type   | Map            | Extents | Info |
|------|--------|----------------|---------|------|
| a    | Array  | P T F TF A C U | (:,:,:) |      |
| b    | Array  | P T F TF A C U | (:,:,:) |      |
| bnd  | Array  | P T F TF A C U | (:,:,:) |      |
| c    | Array  | P T F TF A C U | (:,:,:) |      |
| i    | Scalar | P T F TF A C U |         |      |
| imax | Scalar | P T F TF A C U |         |      |
| j    | Scalar | P T F TF A C U |         |      |
| jmax | Scalar | P T F TF A C U |         |      |
| k    | Scalar | P T F TF A C U |         |      |
| kmax | Scalar | P T F TF A C U |         |      |

First/Last Private

Enable FirstPrivate

Enable LastPrivate

Map Always

Always

Metadirective

Metadirective

Collapse

Reduction

None

Find Name:

Insert Directive

Show Directive

Close

A loop starting at line 211 was scoped without errors.

A loop starting at line 211 is flat (contains no external calls).

A loop starting at line 211 was not vectorized because a better candidate was found at line 213.

himenopl loaded. himenox+508124-4242835t loaded.



## Navigation

◀ Loop Performance ▶ ⚙

- ▶ 63.1397 JACOBI@208
- ▶ 62.6147 JACOBI@211 (gpu) ■
- ▶ 0.2107 JACOBI@232
- ▶ 0.1961 INITMT@154
- ▶ 0.1960 INITMT@155
- ▶ 0.1871 INITMT@156
- ▶ 0.1576 INITMT@175
- ▶ 0.1039 JACOBI@233

## Source - /lvol/levesque/himeno\_hybrid/himenoBMTxpr.f

cce/14.0.1

Up

Down

Save



```
+
! Directive inserted by Cray Reveal. May be incomplete.
!$OMP target teams distribute
!$OMP& private(i, j, k, s0, ss)
!$OMP& firstprivate(imax, jmax, kmax, omega)
!$OMP& map(tofrom: wgosa)
!$OMP& reduction(+: wgosa)
!$OMP& map(always, to: a(:,:,:), b(:,:,:), bnd(:,:,:), c(:,:,:),
!$OMP& p(:,:,:), wrk1(:,:,:))
!$OMP& map(always, tofrom: wrk2(:,:,:))
FS 211 D0 K=2,kmax-1
  F 212 D0 J=2,jmax-1
    FVr3 213 D0 I=2,imax-1
      214 S0=a(I,J,K,1)*p(I+1,J,K)+a(I,J,K,2)*p(I,J+1,K)
      215 1 +a(I,J,K,3)*p(I,J,K+1)
      216 2 +b(I,J,K,1)*(p(I+1,J+1,K)-p(I+1,J-1,K)
      217 3 -p(I-1,J+1,K)+p(I-1,J-1,K))
      218 4 +b(I,J,K,2)*(p(I,J+1,K+1)-p(I,J-1,K+1)
      219 5 -p(I,J+1,K-1)+p(I,J-1,K-1))
      220 6 +b(I,J,K,3)*(p(I+1,J,K+1)-p(I-1,J,K+1)
      221 7 -p(I+1,J,K-1)+p(I-1,J,K-1))
      222 8 +c(I,J,K,1)*p(I-1,J,K)+c(I,J,K,2)*p(I,J-1,K)
      223 9 +c(I,J,K,3)*p(I,J,K-1)+wrk1(I,J,K)
      224 SS=(S0*a(I,J,K,4)-p(I,J,K))*bnd(I,J,K)
      225 WGOSA=WGOSA+SS*SS
      226 wrk2(I,J,K)=p(I,J,K)+OMEGA *SS
      227 enddo
      228 enddo
      229 enddo
      230 C
```

## Info - Line 211

- A loop starting at line 211 was scoped without errors.
- A loop starting at line 211 is flat (contains no external calls).
- A loop starting at line 211 was not vectorized because a better candidate was found at line 213.

# Commands for using perftools

- `module swap perftools-lite-loops perftools`
- `ftn -hlist=a -homp himenoBMTxpr.f -o himenox`
- `pat_build -u -g mpi himenox`
- `srun -n 2 /himenox+pat`
- `pat_report -T <directory generated by execution>`



Table 1: Profile by Function Group and Function

| Time%  | Time       | Imb. Time | Imb.  | Calls     | Group                           |
|--------|------------|-----------|-------|-----------|---------------------------------|
|        |            |           | Time% |           | Function                        |
|        |            |           |       |           | PE=HIDE                         |
|        |            |           |       |           | Thread=HIDE                     |
| 100.0% | 517.005427 | --        | --    | 170,510.0 | Total                           |
| -----  |            |           |       |           |                                 |
| 93.2%  | 482.005140 | --        | --    | 40,012.0  | OACC                            |
| -----  |            |           |       |           |                                 |
| 86.8%  | 448.955431 | 20.973100 | 6.0%  | 10,000.0  | jacobi_.ACC_COPY@li.213         |
| 5.5%   | 28.517275  | 2.813567  | 12.0% | 10,000.0  | jacobi_.ACC_COPY@li.239         |
| 0.5%   | 2.609518   | 0.576732  | 24.1% | 10,000.0  | jacobi_.ACC_SYNC_WAIT@li.239    |
| 0.2%   | 1.233697   | 0.009083  | 1.0%  | 6.0       | jacobi_.ACC_COPY@li.89          |
| 0.1%   | 0.687733   | 0.134373  | 21.8% | 10,000.0  | jacobi_.ACC_ASYNC_KERNEL@li.213 |
| 0.0%   | 0.000823   | 0.000119  | 16.8% | 3.0       | jacobi_.ACC_ASYNC_KERNEL@li.89  |
| 0.0%   | 0.000663   | 0.000142  | 23.4% | 3.0       | jacobi_.ACC_SYNC_WAIT@li.89     |
| =====  |            |           |       |           |                                 |
| 3.9%   | 20.382070  | --        | --    | 110,053.0 | MPI                             |
| -----  |            |           |       |           |                                 |
| 3.3%   | 17.007617  | 13.779150 | 59.7% | 20,006.0  | MPI_WAITALL                     |
| 0.5%   | 2.339848   | 1.328698  | 48.3% | 40,012.0  | MPI_ISEND                       |
| 0.2%   | 0.984848   | 0.743195  | 57.3% | 40,012.0  | MPI_IRecv                       |
| 0.0%   | 0.049582   | 0.002511  | 6.4%  | 10,004.0  | MPI_ALLREDUCE                   |
| 0.0%   | 0.000066   | 0.000003  | 5.3%  | 1.0       | MPI_CART_CREATE                 |
| 0.0%   | 0.000032   | 0.000005  | 17.7% | 1.0       | MPI_REDUCE                      |
| 0.0%   | 0.000027   | 0.000004  | 16.0% | 2.0       | MPI_BARRIER                     |
| 0.0%   | 0.000021   | 0.000002  | 10.9% | 4.0       | MPI_WTIME                       |
| 0.0%   | 0.000008   | 0.000006  | 52.7% | 2.0       | MPI_TYPE_COMMIT                 |
| 0.0%   | 0.000008   | 0.000001  | 20.2% | 2.0       | MPI_TYPE_VECTOR                 |
| 0.0%   | 0.000005   | 0.000000  | 3.9%  | 1.0       | MPI_CART_GET                    |
| 0.0%   | 0.000003   | 0.000000  | 0.7%  | 2.0       | MPI_CART_SHIFT                  |

```

205.      C
206.      include 'mpif.h'
207.      include 'param.h'
208.      C
209. + 1-----<      DO loop=1,nn
210.   1      gosa=0.0
211.   1      wgos=0.0
212.   1      ! Directive inserted by Cray Reveal. May be incomplete.
213. + 1 MG-----< !$OMP target teams distribute
214.   1 MG      !$OMP& private(i, j, k, s0, ss)
215.   1 MG      !$OMP& firstprivate(imax, jmax, kmax, omega)
216.   1 MG      !$OMP& map(tofrom: wgos)
217.   1 MG      !$OMP& reduction(+: wgos)
218.   1 MG      !$OMP& map(always, to: a(:,:,:), b(:,:,:), bnd(:,:,:), c(:,:,:),
219.   1 MG      !$OMP& p(:,:,:), wrk1(:,:,:))
220.   1 MG      !$OMP& map(always, tofrom: wrk2(:,:,:))
221.   1 MG g-----<      DO K=2, kmax-1
222. + 1 MG g 4-----<      DO J=2, jmax-1
223.   1 MG g 4 gr3---<      DO I=2, imax-1
224.   1 MG g 4 gr3      S0=a(I, J, K, 1)*p(I+1, J, K)+a(I, J, K, 2)*p(I, J+1, K)
225.   1 MG g 4 gr3      1      +a(I, J, K, 3)*p(I, J, K+1)
226.   1 MG g 4 gr3      2      +b(I, J, K, 1)*(p(I+1, J+1, K)-p(I+1, J-1, K)
227.   1 MG g 4 gr3      3      -p(I-1, J+1, K)+p(I-1, J-1, K))
228.   1 MG g 4 gr3      4      +b(I, J, K, 2)*(p(I, J+1, K+1)-p(I, J-1, K+1)
229.   1 MG g 4 gr3      5      -p(I, J+1, K-1)+p(I, J-1, K-1))
230.   1 MG g 4 gr3      6      +b(I, J, K, 3)*(p(I+1, J, K+1)-p(I-1, J, K+1)
231.   1 MG g 4 gr3      7      -p(I+1, J, K-1)+p(I-1, J, K-1))
232.   1 MG g 4 gr3      8      +c(I, J, K, 1)*p(I-1, J, K)+c(I, J, K, 2)*p(I, J-1, K)
233.   1 MG g 4 gr3      9      +c(I, J, K, 3)*p(I, J, K-1)+wrk1(I, J, K)
234.   1 MG g 4 gr3      SS=(S0*a(I, J, K, 4)-p(I, J, K))*bnd(I, J, K)
235.   1 MG g 4 gr3      WGOSA=WGOSA+SS*SS
236.   1 MG g 4 gr3      wrk2(I, J, K)=wrk1(I, J, K)+OMEGA *SS
237.   1 MG g 4 gr3--->      enddo
238.   1 MG g----->      enddo
239.   1 MG g----->>      enddo

```

```

240.      1                      C
241.    + 1 2-----<          DO K=2,kmax-1
242.    + 1 2 3-----<          DO J=2,jmax-1
243.      1 2 3 A-----<          DO I=2,imax-1
244.      1 2 3 A              p(I,J,K)=wrk2(I,J,K)
245.      1 2 3 A----->          enddo
246.      1 2 3----->          enddo
247.      1 2----->          enddo
248.      1                      C
249.    + 1 E-----<>        call sendp(ndx,ndy,ndz). ! P halo exchange
250.      1                      C
251.    + 1                      call mpi_allreduce(wgosa,
252.      1                      >          gosa,
253.      1                      >          1,
254.      1                      >          mpi_real4,
255.      1                      >          mpi_sum,
256.      1                      >          mpi_comm_world,
257.      1                      >          ierr)
258.      1                      C
259.      1----->          enddo
260.                      CC End of iteration

```



Table 1: Profile by Function Group and Function

| Time%  | Time     | Imb.<br>Time | Imb.<br>Time% | Calls   | Group<br>Function<br>PE=HIDE<br>Thread=HIDE |
|--------|----------|--------------|---------------|---------|---------------------------------------------|
| 100.0% | 6.289939 | --           | --            | 2,091.0 | Total                                       |
| 72.5%  | 4.558013 | --           | --            | 384.0   | OACC                                        |
| 55.7%  | 3.502276 | 0.043724     | 1.6%          | 93.0    | jacobi_.ACC_COPY@li.212                     |
| 12.5%  | 0.787380 | 0.021668     | 3.6%          | 6.0     | jacobi_.ACC_COPY@li.89                      |
| 3.8%   | 0.236862 | 0.005840     | 3.2%          | 93.0    | jacobi_.ACC_COPY@li.238                     |
| 0.4%   | 0.023946 | 0.005695     | 25.6%         | 93.0    | jacobi_.ACC_SYNC_WAIT@li.238                |
| 0.1%   | 0.006408 | 0.001959     | 31.2%         | 93.0    | jacobi_.ACC_ASYNC_KERNEL@li.212             |
| 0.0%   | 0.000692 | 0.000102     | 17.2%         | 3.0     | jacobi_.ACC_SYNC_WAIT@li.89                 |
| 0.0%   | 0.000450 | 0.000134     | 30.6%         | 3.0     | jacobi_.ACC_ASYNC_KERNEL@li.89              |
| 21.5%  | 1.353529 | --           | --            | 99.0    | USER                                        |
| 18.0%  | 1.129580 | 0.024657     | 2.8%          | 1.0     | himenobmtxp_                                |
| 2.0%   | 0.124014 | 0.008535     | 8.6%          | 1.0     | initmt_                                     |
| 1.6%   | 0.098615 | 0.016419     | 19.0%         | 1.0     | jacobi_                                     |
| 0.0%   | 0.001281 | 0.000173     | 15.8%         | 93.0    | jacobi_.ACC_REGION@li.212                   |
| 0.0%   | 0.000039 | 0.000009     | 25.2%         | 3.0     | jacobi_.ACC_REGION@li.89                    |
| 4.4%   | 0.277157 | --           | --            | 430.0   | ETC                                         |
| 4.4%   | 0.276967 | 0.000226     | 0.1%          | 17.0    | _STOP3                                      |
| 0.0%   | 0.000121 | 0.000013     | 12.6%         | 10.0    | cuEGLApiInit                                |
| 0.0%   | 0.000057 | 0.000048     | 60.8%         | 400.0   | __libc_csu_init                             |
| 0.0%   | 0.000008 | 0.000001     | 15.5%         | 1.0     | cudbgMain                                   |
| 0.0%   | 0.000005 | 0.000001     | 19.9%         | 2.0     | cuVDPACtxCreate                             |
| 1.5%   | 0.091650 | --           | --            | 1,076.0 | MPI                                         |
| 1.1%   | 0.071594 | 0.054624     | 57.7%         | 192.0   | MPI_WAITALL                                 |
| 0.2%   | 0.014993 | 0.002644     | 20.0%         | 384.0   | MPI_ISEND                                   |
| 0.1%   | 0.004539 | 0.000897     | 22.0%         | 384.0   | MPI_IRECV                                   |
| 0.0%   | 0.000397 | 0.000015     | 4.9%          | 97.0    | MPI_ALLREDUCE                               |
| 0.0%   | 0.000053 | 0.000003     | 6.7%          | 1.0     | MPI_CART_CREATE                             |

# Complications with Himeno

- All Arrays are in COMMON blocks – need to be global
- Reveal is putting data transfer inside the iteration loop
  - However, it does recognize that on P and WRK2 need to be updated
- Biggest issue is transferring the entire P array when only boundaries need to be exchanged
- Following charts give some optimizations for himeno that do not deal with halo exchange of P
  - The plan is to put all the arrays and operations on those arrays on the accelerator.
  - Whenever the host needs to do something with the accelerator data, we must update the host from the accelerator



# Intialize arrays on the accelerator

```
148.      C*****
149.      subroutine initmt(mz,it)
150.      C*****
151.      IMPLICIT REAL*4(a-h,o-z)
152.      C
153.      include 'param.h'
154.      C
155.      + !$OMP TARGET DATA map(alloc:a,b,c,bnd,wrk1,wrk2,p)
156.      + MG-----< !$OMP target teams distribute
157.      MG g-----< do k=1,mkmax
158.      + MG g C-----< do j=1,mjmax
159.      MG g C gCr2-----< do i=1,mimax
160.      MG g C gCr2 a(i,j,k,1)=0.0
161.      MG g C gCr2 a(i,j,k,2)=0.0
162.      MG g C gCr2 a(i,j,k,3)=0.0
163.      MG g C gCr2 a(i,j,k,4)=0.0
164.      MG g C gCr2 b(i,j,k,1)=0.0
165.      MG g C gCr2 b(i,j,k,2)=0.0
166.      MG g C gCr2 b(i,j,k,3)=0.0
167.      MG g C gCr2 c(i,j,k,1)=0.0
168.      MG g C gCr2 c(i,j,k,2)=0.0
169.      MG g C gCr2 c(i,j,k,3)=0.0
170.      MG g C gCr2 A--<> p(i,j,k)=0.0
171.      MG g C gCr2 A--<> wrk1(i,j,k)=0.0
172.      MG g C gCr2 A--<> wrk2(i,j,k)=0.0
173.      MG g C gCr2 A--<> bnd(i,j,k)=0.0
174.      MG g C gCr2-----> enddo
175.      MG g C-----> enddo
176.      MG g----->> enddo
```

```

205.      C
206.      subroutine jacobi(nn, gosa)
207.      C*****
208.      IMPLICIT REAL*4(a-h, o-z)
209.      C
210.      include 'mpif.h'
211.      include 'param.h'
212.      C
213.      + 1-----<      DO loop=1, nn
214.      1      gosa=0.0
215.      1      wgosa=0.0
216.      + 1      !$OMP TARGET DATA map(tofrom:p)
217.      1      ! Directive inserted by Cray Reveal. May be incomplete.
218.      + 1 MG-----< !$OMP target teams distribute
219.      1 MG      !$OMP& private(i, j, k, s0, ss)
220.      1 MG      !$OMP& firstprivate(imax, jmax, kmax, omega)
221.      1 MG      !$OMP& map(tofrom: wgosa)
222.      1 MG      !$OMP& reduction(+: wgosa)
223.      1 MG g-----<      DO K=2, kmax-1
224.      + 1 MG g 4-----<      DO J=2, jmax-1
225.      1 MG g 4 gr3----<      DO I=2, imax-1
226.      1 MG g 4 gr3      S0=a(I, J, K, 1)*p(I+1, J, K)+a(I, J, K, 2)*p(I, J+1, K)
227.      1 MG g 4 gr3      1      +a(I, J, K, 3)*p(I, J, K+1)
228.      1 MG g 4 gr3      2      +b(I, J, K, 1)*(p(I+1, J+1, K)-p(I+1, J-1, K)
229.      1 MG g 4 gr3      3      -p(I-1, J+1, K)+p(I-1, J-1, K))
230.      1 MG g 4 gr3      4      +b(I, J, K, 2)*(p(I, J+1, K+1)-p(I, J-1, K+1)
231.      1 MG g 4 gr3      5      -p(I, J+1, K-1)+p(I, J-1, K-1))
232.      1 MG g 4 gr3      6      +b(I, J, K, 3)*(p(I+1, J, K+1)-p(I-1, J, K+1)
233.      1 MG g 4 gr3      7      -p(I+1, J, K-1)+p(I-1, J, K-1))
234.      1 MG g 4 gr3      8      +c(I, J, K, 1)*p(I-1, J, K)+c(I, J, K, 2)*p(I, J-1, K)
235.      1 MG g 4 gr3      9      +c(I, J, K, 3)*p(I, J, K-1)+wrk1(I, J, K)
236.      1 MG g 4 gr3      SS=(S0*a(I, J, K, 4)-p(I, J, K))*bnd(I, J, K)
237.      1 MG g 4 gr3      WGOSA=WGOSA+SS*SS
238.      1 MG g 4 gr3      wrk2(I, J, K)=p(I, J, K)+OMEGA *SS
239.      1 MG g 4 gr3      r3----->      enddo
240.      1 MG g 4----->      enddo
241.      1 MG g----->>      enddo

```

```

242.      1              C
243. + 1 MG-----< !$OMP target teams distribute
244.      1 MG          !$OMP& private(i, j, k)
245.      1 MG          !$OMP& firstprivate(imax, jmax, kmax)
246.      1 MG g-----<          DO K=2,kmax-1
247. + 1 MG g 4-----<          DO J=2,jmax-1
248.      1 MG g 4 g-----<          DO I=2,imax-1
249.      1 MG g 4 g          p(I, J, K)=wrk2(I, J, K)
250.      1 MG g 4 g----->          enddo
251.      1 MG g 4----->          enddo
252.      1 MG g----->>          enddo
253.      1              C
254.      1              !$OMP END TARGET DATA
255. + 1 E-----<>          call sendp(ndx, ndy, ndz)
256.      1              C
257. + 1              call mpi_allreduce(wgosa,
258.      1              >          gosa,
259.      1              >          1,
260.      1              >          mpi_real4,
261.      1              >          mpi_sum,
262.      1              >          mpi_comm_world,
263.      1              >          ierr)
264.      1              C
265.      1----->          enddo
266.              CC End of iteration
267.              return
268.              end

```



# At this point I am moving to illustrate halo exchanges on a different application – leslie3d

- Himeno uses MPI message types which are more confusing
- Leslie3d packs and unpacks buffers and it is easier to illustrate optimizing on the GPU



# One of the eight buffers required for 3-D application

```
IF(NCY .LT. NPY .OR. (NCY .EQ. NPY .AND. JPERIODIC)) THEN
  JCNT = 0
  DO K = 1-NLEVELS, KCMAX+NLEVELS
    DO J = JCMAX-NLEVELS+1, JCMAX
      DO I = 1-NLEVELS, ICMAX+NLEVELS
        JCNT = JCNT + 1
        S_N(JCNT) = VAR(I,J,K)
      END DO
    ENDDO
  END DO
  CALL MPI_SEND(S_N,
>               NLEVELS * NSIZE1 * NSIZE3,
>               MPI_DOUBLE_PRECISION,
>               NORTH,
>               MSGFLAG + MSGTSN,
>               MPI_COMM_WORLD,
>               IERR )
  IF ( IERR .NE. 0 ) CALL EJECT ('MPI_SEND FAILED: S_N')
END IF
```

Need to pack buffers on  
the accelerator

# Packing on the GPU and sending the halo

We do have to update  
the host prior to the  
MPI\_SEND

```
738.      #ifdef OMP_TARGET
739.      G-----< !$omp target teams distribute
740.      G      #endif
741.      G g-----<          DO K = 1, NLEVELS
742.      + G g 3-----<          DO J = 1-NLEVELS, JCMAX+NLEVELS
743.      G g 3      #ifdef OMP_TARGET
744.      + G g 3      !$omp parallel do private(kcnt) private(i)
745.      G g 3      #endif
746.      G g 3 g-----<          DO I = 1-NLEVELS, ICMAX+NLEVELS
747.      G g 3 g      KCNT = (k-1)*(JCMAX+2*NLEVELS)*(ICMAX+2*NLEVELS)
748.      G g 3 g      >          +(j-1+NLEVELS)*(ICMAX+2*NLEVELS)+i
749.      G g 3 g      S_I(KCNT) = VAR(I,J,K)
750.      G g 3 g----->          END DO
751.      G g 3----->          ENDDO
752.      G g----->          END DO
753.      G      #ifdef OMP_TARGET
754.      G-----> !$omp end target teams distribute
755.      !$omp target update from(S_I)
756.      #endif
757.      +      CALL MPI_SEND(S_I,
758.      >          NLEVELS * NSIZE1 * NSIZE2,
759.      >          MPI_DOUBLE_PRECISION,
760.      >          IN,
761.      >          MSGFLAG + MSGTSI,
762.      >          MPI_COMM_WORLD,
763.      >          IERR )
764.      +      IF ( IERR .NE. 0 ) CALL EJECT ('MPI_SEND FAILED: S_I')
765.      END IF
```

# Posting the receive on the Host

We don't have to do anything prior to or after the IRECV

```
706.
707.      IF(NCZ .GT.  1 .OR. (NCZ .EQ.  1 .AND. KPERIODIC)) THEN
708.  +      CALL MPI_Irecv(R_I,
709.      >      NLEVELS * NSIZE1 * NSIZE2,
710.      >      MPI_DOUBLE_PRECISION,
711.      >      IN,
712.      >      MSGFLAG + MSGTRI,
713.      >      MPI_COMM_WORLD,
714.      >      MSGIDRI,
715.      >      IERR )
```



# Receiving and unpacking on the GPU

```
785. +          CALL MPI_WAIT( MSGIDRI, MPI_STATUS, IERR)
786.
787.          #ifdef OMP_TARGET
788.          !$omp target update to(R_I)
789.          G-----< !$omp target teams distribute
790.          G          #endif
791.          G g-----<          DO K = 1-NLEVELS, 0
792.          + G g 3-----<          DO J = 1-NLEVELS, JCMAX+NLEVELS
793.          G g 3          #ifdef OMP_TARGET
794.          + G g 3          !$omp parallel do private(kcnt) private(i)
795.          G g 3          #endif
796.          G g 3 g-----<          DO I = 1-NLEVELS, ICMAX+NLEVELS
797.          G g 3 g          KCNT = (k-1+NLEVELS)*(JCMAX+2*NLEVELS)*(ICMAX+2*NLEVELS)
798.          G g 3 g          >          +(j-1+NLEVELS)*(ICMAX+2*NLEVELS)+i
799.          G g 3 g          VAR(I,J,K) = R_I(KCNT)
800.          G g 3 g----->          END DO
801.          G g 3----->          END DO
802.          G g----->          END DO
803.          G          #ifdef OMP_TARGET
804.          G-----> !$omp end target teams distribute
805.          #endif
```

After the MPI\_WAIT we  
need to update the GPU  
with the communicated  
array



# Still can do better – use use\_device\_ptr and ~~setenv~~ MPICH\_RDMA\_ENABLED\_CUDA 1

```
#ifdef OMP_TARGET
!$omp target enter data map(to:VAR)
!$omp&      map(alloc:R_N,R_W,R_S,R_E,R_I,R_O,S_N,S_W,S_S,
!$OMP&      S_E,S_I,S_O)
!$omp target data
!$omp& use_device_ptr(R_N,R_W,R_S,R_E,R_I,R_O,S_N,S_W,S_S,
!$OMP&      S_E,S_I,S_O)
#endif
```

```
CALL MPI_Irecv(R_W,
>      NLEVELS * NSIZE2 * NSIZE3,
>      MPI_DOUBLE_PRECISION,
>      WEST,
>      MSGFLAG + MSGTRW,
>      MPI_COMM_WORLD,
>      MSGIDRW,
>      IERR)
```

Much easier to code –  
Do not need target  
update clauses

# Still can do better

```
ifdef OMP_TARGET
!$omp target enter data map(to:VAR)
!$omp&      map(alloc:R_N,R_W,R_S,R_E,R_I,R_O,S_N,S_W,S_S,
!$OMP&      S_E,S_I,S_O)
!$omp target data
!$omp& use_device_ptr(R_N,R_W,R_S,R_E,R_I,R_O,S_N,S_W,S_S,
!$OMP&      S_E,S_I,S_O)
#endif
```

```
CALL MPI_SEND(S_E,
>      NLEVELS * NSIZE2 * NSIZE3,
>      MPI_DOUBLE_PRECISION,
>      EAST,
>      MSGFLAG + MSGTSE,
>      MPI_COMM_WORLD,
>      IERR )
```

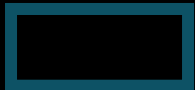
Much easier to code



# All Packing done on the GPU

# CRAY\_ACC\_DEBUG Environment Variable

- Three levels of verbosity
  - 1 High Level overview of kernels executed and data transferred
  - 2 Breaks down data transfer by each variable
  - 3 The whole Kitchen sink



# setenv CRAY\_ACC\_DEBUG 1

```
ACC: Transfer 29 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:21
ACC: Transfer 15 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:114
ACC: Transfer 1 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:141
ACC: Transfer 29 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:21
ACC: Transfer 15 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:114
ACC: Transfer 1 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:141
ACC: Execute kernel extrapi_$ck_L141_7 async(auto) from src/fluxi.f:141
ACC: Execute kernel extrapi_$ck_L141_7 async(auto) from src/fluxi.f:141
ACC: Wait async(auto) from src/fluxi.f:172
ACC: Transfer 1 items (to acc 0 bytes, to host 0 bytes) from src/fluxi.f:172
ACC: Execute kernel extrapi_$ck_L173_9 async(auto) from src/fluxi.f:173
ACC: Wait async(auto) from src/fluxi.f:172
```



# setenv CRAY\_ACC\_DEBUG 2

```
ACC: Start transfer 29 items from src/fluxi.f:21
ACC:      present 'dq' (255923712 bytes)
ACC:      present 'ds' (95971392 bytes)
ACC:      present 'ds1' (95971392 bytes)
ACC:      present 'dtv' (31990464 bytes)
ACC:      present 'hf' (127961856 bytes)
ACC:      present 'pav' (31990464 bytes)
ACC:      present 'q' (511847424 bytes)
ACC:      present 'qav' (255923712 bytes)
ACC:      present 'six' (31990464 bytes)
ACC:      present 'siy' (31990464 bytes)
ACC:      present 'siz' (31990464 bytes)
ACC:      present 't' (31990464 bytes)
```

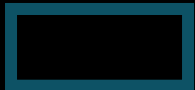
# setenv CRAY\_ACC\_DEBUG 3

```
ACC: Start transfer 29 items from src/fluxi.f:21
ACC:   flags: RETURN_ACC_TIME
ACC:
ACC:   Trans 1
ACC:     Simple transfer of 'dq' (144 bytes)
ACC:       host ptr 78b538
ACC:       acc  ptr 0
ACC:       flags: DOPE_VECTOR DV_ONLY_DATA ALLOCATE COPY_HOST_TO_ACC ACQ_PRESENT REG_PRESENT
ACC:       Transferring dope vector
ACC:         dim:1 lowbound:-2 extent:198 stride_mult:1
ACC:         dim:2 lowbound:-2 extent:198 stride_mult:198
ACC:         dim:3 lowbound:-2 extent:102 stride_mult:39204
ACC:         dim:4 lowbound:1 extent:8 stride_mult:3998808
ACC:         DV size=255923712 (scale:8, elem_size:8)
ACC:         total mem size=255923712 (dv:0 obj:255923712)
ACC:         host region 4dac8780 to 5ced9d80 found in present table index 0 (ref count 2)
ACC:         memory found in present table (2aaaf2000000, base 2aaaf2000000)
ACC:         new acc ptr 2aaaf2000000
```



# So what have we learned

- Perftools is excellent for identifying issues in existing applications for improving threading, vectorization and scalar optimization
- Reveal can help with the difficult job of scoping variables in potential parallelizable loops
  - More difficult if not impossible with C++
- Moving to the GPU is difficult; however, it can be done in steps that are more manageable
  - Perftools identifies the bottlenecks in the GPU application very quickly
- GPU direct is best way to do the message passing – in this case it only matters at scale



# Thank you

