Methodology for Optimizing an application for Massively Parallel Processors

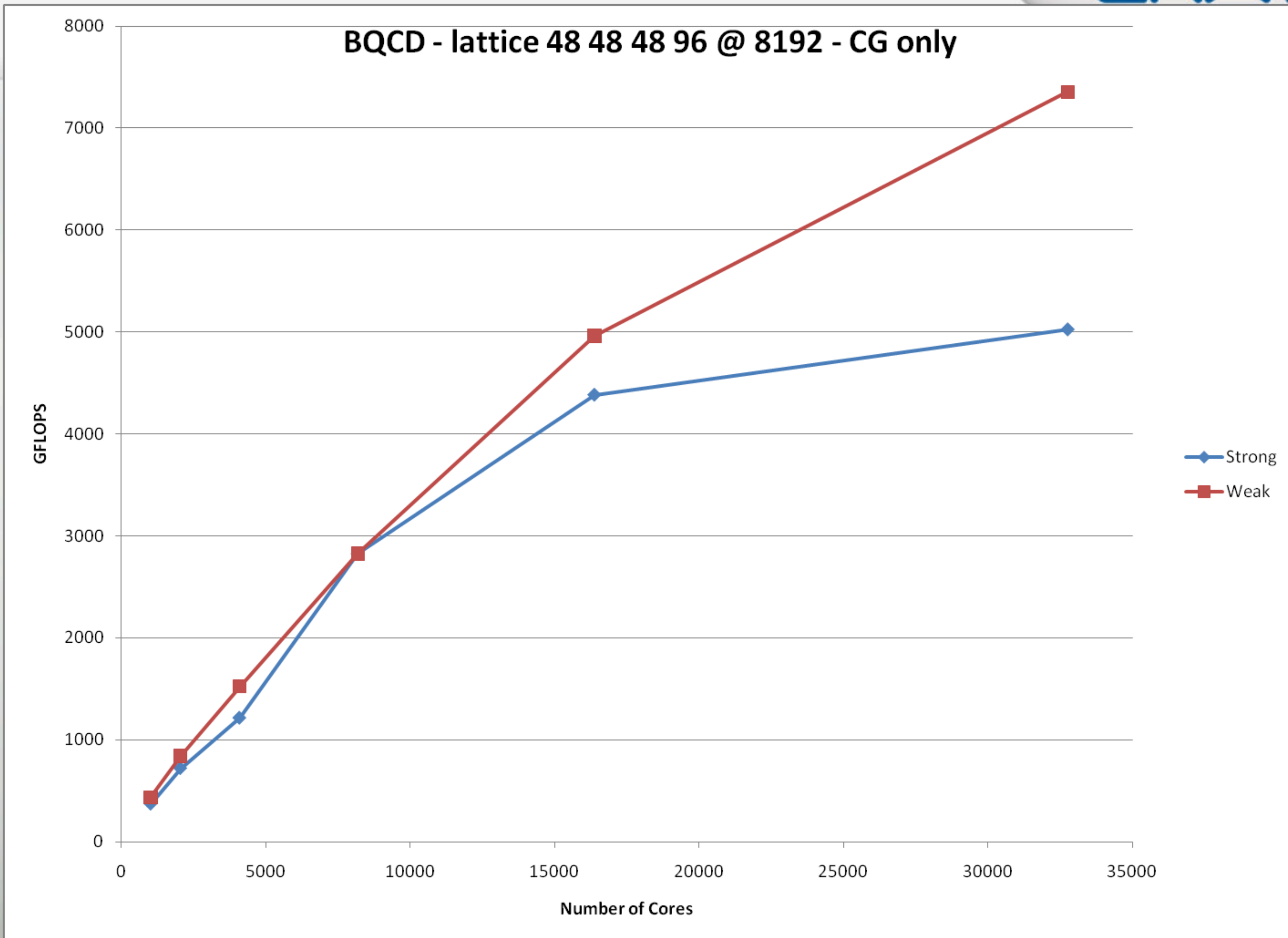
John Levesque

February, 2012

The steps – 1) Identify Application and Science Worthy Problem

- Formulate the problem
  - The problem identified should make good science sense
    - No publicity stunts that are not of interest
  - It should be a production style problem
    - Weak scaling
      - Finer grid as processors increase
      - Fixed amount of work when processors increase
    - Strong scaling
      - Fixed problem size as processors increase
        - Less and less work for each processor as processors increase

Think Bigger


BQCD - lattice 48 48 48 96 @ 8192 - CG only
GFLOPS
0
1000
2000
3000
4000
5000
6000
7000
8000
0
5000
10000
15000
20000
25000
30000
35000
Number of Cores
Strong
Weak

The steps – 2) Instrument the application

- Instrument the application
  - Run the production case
    - Run long enough that the initialization does not use > 1% of the time
    - Run with normal I/O
  - Use Craypat's APA
    - First gather sampling for line number profile
    - Second gather instrumentation (-g mpi,io)
      - Hardware counters
      - MPI message passing information
      - I/O information

load module
make
pat_build -O apa a.out
Execute
pat_report *.xf
pat_build –O *.apa
Execute

Using Craypat MPI statistics

```
MPI Msg Bytes |  MPI Msg |   MsgSz | 16B<= |  256B<= |   4KB<= |Experiment=1
              |    Count |    <16B | MsgSz |   MsgSz |   MsgSz |Function
              |          |   Count | <256B |    <4KB |   <64KB | Caller
              |          |         | Count |   Count |   Count |  PE[mmm]
 3062457144.0 | 144952.0 | 15022.0 |  39.0 | 64522.0 | 65369.0 |Total
|---------------------------------------------------------------------------
| 3059984152.0 | 129926.0 |      -- |  36.0 | 64522.0 | 65368.0 |mpi_isend_
||--------------------------------------------------------------------------
|| 1727628971.0 |  63645.1 |      -- |   4.0 | 31817.1 | 31824.0 |MPP_DO_UPDATE_R8_3DV.in.MPP_DOMAINS_MOD
3|              |          |         |       |         |         | MPP_UPDATE_DOMAIN2D_R8_3DV.in.MPP_DOMAINS_MOD
||||------------------------------------------------------------------------
4||| 1680716892.0 |  61909.4 |      -- |    -- | 30949.4 | 30960.0 |DYN_CORE.in.DYN_CORE_MOD
5|||              |          |         |       |         |         | FV_DYNAMICS.in.FV_DYNAMICS_MOD
6|||              |          |         |       |         |         |  ATMOSPHERE.in.ATMOSPHERE_MOD
7|||              |          |         |       |         |         |   MAIN__
8|||              |          |         |       |         |         |    main
|||||||||-------------------------------------------------------------------
9|||||||| 1680756480.0 |  61920.0 |      -- |    -- | 30960.0 | 30960.0 |pe.13666
9|||||||| 1680756480.0 |  61920.0 |      -- |    -- | 30960.0 | 30960.0 |pe.8949
9|||||||| 1651777920.0 |  54180.0 |      -- |    -- | 23220.0 | 30960.0 |pe.12549
|||||||||===================================================================
```

Memory allocation data from Craypat

```
Table 7:  Heap Leaks during Main Program

 Tracked | Tracked | Tracked |Experiment=1
  MBytes |  MBytes | Objects |Caller
     Not |     Not |     Not | PE[mmm]
 Freed % |   Freed |   Freed |

  100.0% | 593.479 |   43673 |Total
|-----------------------------------------
|   97.7% | 579.580 |   43493 |_F90_ALLOCATE
||----------------------------------------
||   61.4% | 364.394 |     106 |SET_DOMAIN2D.in.MPP_DOMAINS_MOD
3|         |         |         | MPP_DEFINE_DOMAINS2D.in.MPP_DOMAINS_MOD
4|         |         |         |  MPP_DEFINE_MOSAIC.in.MPP_DOMAINS_MOD
5|         |         |         |   DOMAIN_DECOMP.in.FV_MP_MOD
6|         |         |         |    RUN_SETUP.in.FV_CONTROL_MOD
7|         |         |         |     FV_INIT.in.FV_CONTROL_MOD
8|         |         |         |      ATMOSPHERE_INIT.in.ATMOSPHERE_MOD
9|         |         |         |       ATMOS_MODEL_INIT.in.ATMOS_MODEL
10         |         |         |        MAIN__
11         |         |         |         main
|||||||||||------------------------------
12||||||||||     0.0% | 364.395 |     110 |pe.43
12||||||||||     0.0% | 364.394 |     107 |pe.8181
12||||||||||     0.0% | 364.391 |      88 |pe.1047
```

Craypat load-imbalance data

```
Table 1:  Profile by Function Group and Function

 Time %  |        Time  |  Imb. Time  |   Imb.  |     Calls  |Experiment=1
         |              |             | Time %  |            |Group
         |              |             |         |            | Function
         |              |             |         |            |  PE='HIDE'

 100.0%  | 1061.141647  |          --  |     --  | 3454195.8  |Total
|---------------------------------------------------------------------
|  70.7% |  750.564025  |          --  |     --  |  280169.0  |MPI_SYNC
||--------------------------------------------------------------------
||  45.3% |  480.828018 | 163.575446 |  25.4% |   14653.0 |mpi_barrier_(sync)
||  18.4% |  195.548030 |  33.071062 |  14.5% |  257546.0 |mpi_allreduce_(sync)
||   7.0% |   74.187977 |   5.261545 |   6.6% |    7970.0 |mpi_bcast_(sync)
||====================================================================
|  15.2% |  161.166842  |          --  |     --  | 3174022.8  |MPI
||--------------------------------------------------------------------
||  10.1% |  106.808182 |   8.237162 |   7.2% |  257546.0 |mpi_allreduce_
||   3.2% |   33.841961 | 342.085777 |  91.0% |  755495.8 |mpi_waitall_
||====================================================================
|  14.1% |  149.410781  |          --  |     --  |       4.0  |USER
||--------------------------------------------------------------------
||  14.0% |  148.048597 | 446.124165 |  75.1% |       1.0 |main
|====================================================================
```

Hardware Counters

```
USER / MPP_DO_UPDATE_R8_3DV.in.MPP_DOMAINS_MOD
------------------------------------------------------------------------
  Time%                                               10.2%
  Time                                           49.386043 secs
  Imb.Time                                        1.359548 secs
  Imb.Time%                                            2.7%
  Calls                         167.1 /sec          8176.0 calls
  PAPI_L1_DCM                  10.512M/sec       514376509 misses
  PAPI_TLB_DM                   2.104M/sec       102970863 misses
  PAPI_L1_DCA                 155.710M/sec      7619492785 refs
  PAPI_FP_OPS                                            0 ops
  User time (approx)           48.934 secs    112547914072 cycles   99.1%Time
  Average Time per Call                           0.006040 sec
  CrayPat Overhead : Time       0.0%
  HW FP Ops / User time                                  0 ops   0.0%peak(DP)
  HW FP Ops / WCT
  Computational intensity       0.00 ops/cycle        0.00 ops/ref
  MFLOPS (aggregate)            0.00M/sec
  TLB utilization              74.00 refs/miss       0.145 avg uses
  D1 cache hit,miss ratios     93.2% hits             6.8% misses
  D1 cache utilization (M)     14.81 refs/miss       1.852 avg uses
```

```
Table 2:  Profile by Group, Function, and Line

 Samp % |    Samp |Imb. Samp |   Imb. |Experiment=1
        |         |          | Samp % |Group
        |         |          |        | Function
        |         |          |        |  Source
        |         |          |        |   Line
        |         |          |        |    PE='HIDE'

 100.0% | 103828 |       -- |     -- |Total
|--------------------------------------------------
|  48.9% |  50784 |       -- |     -- |USER
||-------------------------------------------------
||  11.0% |  11468 |       -- |     -- |MPP_DO_UPDATE_R8_3DV.in.MPP_DOMAI
3|        |        |          |        | shared/mpp/include/mpp_do_update
||||-----------------------------------------------
4|||   2.9% |   3056 |   238.53 |   7.2% |line.380
4|||   2.8% |   2875 |   231.97 |   7.5% |line.967
4|||   2.0% |   2071 |   310.19 |  13.0% |line.1028
||||===============================================
```

Using Craypat MPI statistics

```
MPI Msg Bytes |  MPI Msg |   MsgSz | 16B<= |  256B<= |   4KB<= |Experiment=1
              |    Count |    <16B | MsgSz |   MsgSz |   MsgSz |Function
              |          |   Count | <256B |    <4KB |   <64KB | Caller
              |          |         | Count |   Count |   Count |  PE[mmm]
 3062457144.0 | 144952.0 | 15022.0 |  39.0 | 64522.0 | 65369.0 |Total
|------------------------------------------------------------------------
| 3059984152.0 | 129926.0 |      -- |  36.0 | 64522.0 | 65368.0 |mpi_isend_
||-----------------------------------------------------------------------
|| 1727628971.0 |  63645.1 |      -- |   4.0 | 31817.1 | 31824.0 |MPP_DO_UPDATE_R8_3DV.in.MPP_DOMAINS_MOD
3|              |          |         |       |         |         | MPP_UPDATE_DOMAIN2D_R8_3DV.in.MPP_DOMAINS_MOD
||||--------------------------------------------------------------------
4||| 1680716892.0 |  61909.4 |      -- |    -- | 30949.4 | 30960.0 |DYN_CORE.in.DYN_CORE_MOD
5|||              |          |         |       |         |         | FV_DYNAMICS.in.FV_DYNAMICS_MOD
6|||              |          |         |       |         |         |  ATMOSPHERE.in.ATMOSPHERE_MOD
7|||              |          |         |       |         |         |   MAIN__
8|||              |          |         |       |         |         |    main
|||||||||-------------------------------------------------------------------
9|||||||| 1680756480.0 |  61920.0 |      -- |    -- | 30960.0 | 30960.0 |pe.13666
9|||||||| 1680756480.0 |  61920.0 |      -- |    -- | 30960.0 | 30960.0 |pe.8949
9|||||||| 1651777920.0 |  54180.0 |      -- |    -- | 23220.0 | 30960.0 |pe.12549
|||||||||===================================================================
```

Memory allocation data from Craypat

```
Table 7:  Heap Leaks during Main Program

 Tracked | Tracked | Tracked |Experiment=1
  MBytes |  MBytes | Objects |Caller
     Not |     Not |     Not | PE[mmm]
 Freed % |   Freed |   Freed |

  100.0% | 593.479 |   43673 |Total
|-----------------------------------------
|   97.7% | 579.580 |   43493 |_F90_ALLOCATE
||----------------------------------------
||   61.4% | 364.394 |     106 |SET_DOMAIN2D.in.MPP_DOMAINS_MOD
3|         |         |         | MPP_DEFINE_DOMAINS2D.in.MPP_DOMAINS_MOD
4|         |         |         |  MPP_DEFINE_MOSAIC.in.MPP_DOMAINS_MOD
5|         |         |         |   DOMAIN_DECOMP.in.FV_MP_MOD
6|         |         |         |    RUN_SETUP.in.FV_CONTROL_MOD
7|         |         |         |     FV_INIT.in.FV_CONTROL_MOD
8|         |         |         |      ATMOSPHERE_INIT.in.ATMOSPHERE_MOD
9|         |         |         |       ATMOS_MODEL_INIT.in.ATMOS_MODEL
10         |         |         |        MAIN__
11         |         |         |         main
||||||||||||------------------------------
12||||||||||    0.0% | 364.395 |     110 |pe.43
12||||||||||    0.0% | 364.394 |     107 |pe.8181
12||||||||||    0.0% | 364.391 |      88 |pe.1047
```

Hardware Counters

```
USER / MPP_DO_UPDATE_R8_3DV.in.MPP_DOMAINS_MOD
------------------------------------------------------------------------
  Time%                                             10.2%
  Time                                          49.386043 secs
  Imb.Time                                       1.359548 secs
  Imb.Time%                                          2.7%
  Calls                          167.1 /sec        8176.0 calls
  PAPI_L1_DCM                   10.512M/sec     514376509 misses
  PAPI_TLB_DM                    2.104M/sec     102970863 misses
  PAPI_L1_DCA                  155.710M/sec    7619492785 refs
  PAPI_FP_OPS                                           0 ops
  User time (approx)            48.934 secs  112547914072 cycles  99.1%Time
  Average Time per Call                          0.006040 sec
  CrayPat Overhead : Time    0.0%
  HW FP Ops / User time                                 0 ops  0.0%peak(DP)
  HW FP Ops / WCT
  Computational intensity    0.00 ops/cycle        0.00 ops/ref
  MFLOPS (aggregate)         0.00M/sec
  TLB utilization           74.00 refs/miss       0.145 avg uses
  D1 cache hit,miss ratios  93.2% hits             6.8% misses
  D1 cache utilization (M)  14.81 refs/miss       1.852 avg uses
```

```
Table 2:  Profile by Group, Function, and Line

 Samp % |   Samp |Imb. Samp |   Imb. |Experiment=1
        |        |          | Samp % |Group
        |        |          |        | Function
        |        |          |        |  Source
        |        |          |        |   Line
        |        |          |        |    PE='HIDE'

 100.0% | 103828 |       -- |     -- |Total
|--------------------------------------------------
|  48.9% |  50784 |       -- |     -- |USER
||-------------------------------------------------
||  11.0% |  11468 |       -- |     -- |MPP_DO_UPDATE_R8_3DV.in.MPP_DOMAINS_MOD
3|        |        |          |        | shared/mpp/include/mpp_do_updateV.h
||||-----------------------------------------------
4|||   2.9% |   3056 |   238.53 |   7.2% |line.380
4|||   2.8% |   2875 |   231.97 |   7.5% |line.967
4|||   2.0% |   2071 |   310.19 |  13.0% |line.1028
||||===============================================
```

The steps – 3) Examine Results

- Examine Results
  - Is there load imbalance?
    - Yes – fix it first – go to step 4
    - No – you are lucky
  - Is computation > 50% of the runtime
    - Yes – go to step 5
  - Is communication > 50% of the runtime
    - Yes – go to step 6
  - Is I/O > 50% of the runtime
    - Yes – go to step 7

Always fix load imbalance first

Motivation for Load Imbalance Analysis

- Increasing system software and architecture complexity
  - Current trend in high end computing is to have systems with tens of thousands of processors
    - This is being accentuated with multi-core processors
- Applications have to be very well balanced In order to perform at scale on these MPP systems
  - Efficient application scaling includes a balanced use of requested computing resources
- Desire to minimize computing resource “waste”
  - Identify slower paths through code
  - Identify inefficient “stalls” within an application

Profile with Load Distribution by Groups

```
Table 1:  Profile by Function Group and Function

 Time % |      Time |Imb. Time |   Imb. | Calls |Group
        |           |          | Time % |       | Function
        |           |          |        |       |  PE='HIDE'

 100.0% | 0.482144 |        -- |     -- |  2530 |Total
|-----------------------------------------------------------
|  83.7% | 0.403314 |        -- |     -- |   303 |USER
||----------------------------------------------------------
||  32.4% | 0.156028 | 0.009882 |   6.8% |    98 |calc3_
||  27.7% | 0.133643 | 0.007400 |   6.0% |   100 |calc2_
||  21.0% | 0.101406 | 0.002552 |   2.8% |   100 |calc1_
||   2.0% | 0.009696 | 0.000287 |   3.3% |     1 |inital_
||==========================================================
|  16.3% | 0.078830 |        -- |     -- |  2227 |MPI
||----------------------------------------------------------
||  12.7% | 0.061266 | 0.078133 |  64.1% |   351 |mpi_waitall_
||   2.2% | 0.010607 | 0.011582 |  59.7% |   936 |mpi_isend_
||   1.4% | 0.006945 | 0.004463 |  44.7% |   936 |mpi_irecv_
|===========================================================
```

MPI Sync Time

- Measure load imbalance in programs instrumented to trace MPI functions to determine if MPI ranks arrive at collectives together
- Separates potential load imbalance from data transfer
- Sync times reported by default if MPI functions traced
- If desired, PAT_RT_MPI_SYNC=0 deactivated this feature

MPI Sync Time Statistics

```
 Time %  |      Time  |Imb. Time  |   Imb.  | Calls  |Group
         |            |           | Time %  |        | Function
         |            |           |         |        |  PE='HIDE'

 100.0% | 7.193714 |        -- |      -- | 17604 |Total
|-----------------------------------------------------------
|  76.5% | 5.500078 |        -- |      -- |  4752 |USER
||----------------------------------------------------------
||   96.0% | 5.277791 | 0.171848 |    3.3% |     12 |sweep_
||    3.2% | 0.177352 | 0.005482 |    3.1% |     12 |source_
||    0.3% | 0.018588 | 0.000527 |    2.9% |     12 |flux_err_
||    0.2% | 0.010866 | 0.003033 |   22.8% |   2280 |snd_real_
||    0.1% | 0.005032 | 0.000144 |    2.9% |      1 |initialize_
||    0.1% | 0.004933 | 0.000154 |    3.2% |      1 |initxs_
||    0.1% | 0.002819 | 0.001773 |   40.3% |   2280 |rcv_real_
||==========================================================
|  16.6% | 1.197321 |        -- |      -- |  4603 |MPI
||----------------------------------------------------------
||   93.9% | 1.124227 | 0.277878 |   20.7% |   2280 |mpi_recv_
||    5.9% | 0.070481 | 0.014437 |   17.7% |   2280 |mpi_send_
||    0.2% | 0.002210 | 0.001088 |   34.4% |     32 |mpi_allreduce_
||==========================================================
|   6.3% | 0.453091 |        -- |      -- |    39 |MPI_SYNC
||----------------------------------------------------------
||   61.1% | 0.277012 | 0.215608 |   45.7% |      4 |mpi_bcast_(sync)
||   38.7% | 0.175564 | 0.270049 |   63.2% |     32 |mpi_allreduce_(sync)
||    0.1% | 0.000515 | 0.000265 |   35.5% |      3 |mpi_barrier_(sync)
|===========================================================
```

The steps – 4) Application is load imbalanced

- What is causing the load imbalance
  - Computation
    - Is decomposition appropriate?
    - Would RANK_REORDER help?
  - Communication
    - Is decomposition appropriate?
    - Would RANK_REORDER help?
    - Are receives pre-posted
- OpenMP may help
  - Able to spread workload with less overhead
    - Large amount of work to go from all-MPI to Hybrid
      - Must accept challenge to OpenMP-ize large amount of code
- Go back to step 2
    - Re-gather statistics

Need Craypat reports

Is SYNC time due to computation?

Craypat load-imbalance data

```
Table 1:  Profile by Function Group and Function

 Time %  |        Time |  Imb. Time |   Imb.  |     Calls |Experiment=1
         |             |            | Time %  |           |Group
         |             |            |         |           | Function
         |             |            |         |           |  PE='HIDE'

 100.0%  | 1061.141647 |         -- |      -- | 3454195.8 |Total
|----------------------------------------------------------------------
|  70.7% |  750.564025 |         -- |      -- |  280169.0 |MPI_SYNC
||---------------------------------------------------------------------
||  45.3% |  480.828018 | 163.575446 |  25.4% |   14653.0 |mpi_barrier_(sync)
||  18.4% |  195.548030 |  33.071062 |  14.5% |  257546.0 |mpi_allreduce_(sync)
||   7.0% |   74.187977 |   5.261545 |   6.6% |    7970.0 |mpi_bcast_(sync)
||=====================================================================
|  15.2% |  161.166842 |         -- |      -- | 3174022.8 |MPI
||---------------------------------------------------------------------
||  10.1% |  106.808182 |   8.237162 |   7.2% |  257546.0 |mpi_allreduce_
||   3.2% |   33.841961 | 342.085777 |  91.0% |  755495.8 |mpi_waitall_
||=====================================================================
|  14.1% |  149.410781 |         -- |      -- |       4.0 |USER
||---------------------------------------------------------------------
||  14.0% |  148.048597 | 446.124165 |  75.1% |       1.0 |main
|=============================================================================
```

The steps – 5) Computation is Major Bottleneck

- What is causing the Bottleneck?
  - Computation
    - Is application Vectorized
      - No – vectorize it
    - What library routines are being used?
  - Memory Bandwidth
    - What is cache utilization?
    - TLB problems?
- OpenMP may help
  - Able to spread workload with less overhead
    - Large amount of work to go from all-MPI to Hybrid
      - Must accept challenge to OpenMPize large amount of code
- Go back to step 2
    - Re-gather statistics

Need Hardware counters & Compiler listing in hand

The steps – 6) Communication is Major Bottleneck

- What is causing the Bottleneck?
  - Collectives
    - MPI_ALLTOALL
    - MPI_ALLREDUCE
    - MPI_REDUCE
    - MPI_VGATHER/MPI_VSCATTER
  - Point to Point
    - Are receives pre-posted
      - Don't use MPI_SENDRECV
    - What are the message sizes
      - Small – Combine
      - Large – divide and overlap
- OpenMP may help
  - Able to spread workload with less overhead
    - Large amount of work to go from all-MPI to Hybrid
      - Must accept challenge to OpenMP-ize large amount of code
- Gemini in 2010 will be able to solve current Seastar performance problems

Look at craypat report
MPI message sizes

The steps – 7) I/O is Major Bottleneck

- What type of I/O?
  - One writer – large files
    - Stripe across most OSTs
  - All writers – small files
    - Stripe across one OST
  - MPI-I/O?
    - Try using subset of writers
  - Go back to step 2
    - Re-gather statistics

Look at craypat report on file statistics
Look at read/write sizes


JAGUAR
APPLIED
MECHANICS
Railway
Technical
Research
Institute
H L R I S
KRAKEN

Thank You!