Performance analysis of Sweep3D on Blue Gene/P with the Scalasca toolset

Brian J. N. Wylie*, David Böhme*†, Bernd Mohr*, Zoltán Szebenyi*†, and Felix Wolf*†‡

* *Jülich Supercomputing Centre, Forschungszentrum Jülich, 52425 Jülich, Germany*
† *RWTH Aachen University, 52056 Aachen, Germany*
‡ *German Research School for Simulation Sciences, 52062 Aachen, Germany*
{b.wylie, d.boehme, b.mohr, z.szebenyi, f.wolf}@fz-juelich.de

***Abstract*—In studying the scalability of the Scalasca performance analysis toolset to several hundred thousand MPI processes on IBM Blue Gene/P, we investigated a progressive execution performance deterioration of the well-known ASCI Sweep3D compact application. Scalasca runtime summarization analysis quantified MPI communication time that correlated with computational imbalance, and automated trace analysis confirmed growing amounts of MPI waiting times. Further instrumentation, measurement and analyses pinpointed a conditional section of highly imbalanced computation which amplified waiting times inherent in the associated wavefront communication that seriously degraded overall execution efficiency at very large scales. By employing effective data collation, management and graphical presentation, Scalasca was thereby able to demonstrate performance measurements and analyses with 294,912 processes for the first time.**

***Keywords*- parallel performance measurement & analysis; MPI; scalability of applications & tools;**

I. Introduction

Scalasca is an open-source toolset for scalable performance analysis of large-scale parallel applications [1], [2], [3]. It integrates runtime summarization with automated event trace analysis of MPI and OpenMP applications, for a range of current HPC platforms [4]. Just as the trend for constructing supercomputers from increasing numbers of multicore and manycore processors requires application scalability to exploit them effectively, associated application engineering tools must continually improve their scalability commensurately. To assess the scalability of Scalasca on the *Jugene* IBM Blue Gene/P system [5] at Jülich Supercomputing Centre, consisting of 294,912 (288k) processor cores, we chose to study a highly-scalable compact application.

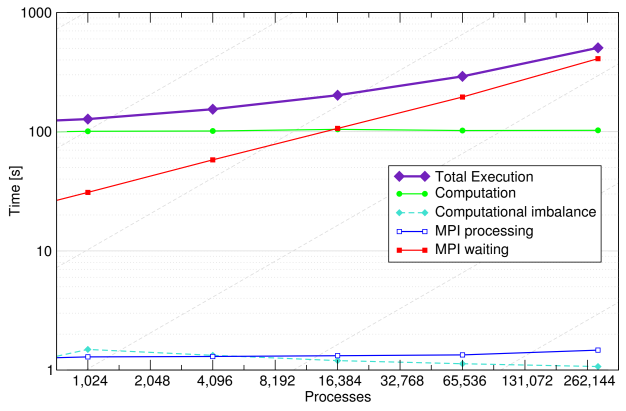
The ASCI Sweep3D benchmark code [6], [7] solves a 1-group time-independent discrete ordinates neutron transport problem, calculating the flux of neutrons through each cell of a three-dimensional grid (i, j, k) along several directions (angles) of travel. Angles are split into eight octants, corresponding to one of the eight directed diagonals of the grid. It uses an explicit two-dimensional decomposition (i, j) of the three-dimensional computation domain, resulting in point-to-point communication of grid-points between neighbouring processes, and reflective boundary conditions. A wavefront process is employed in the i and j directions, combined with pipelining of blocks of k-planes and octants, to expose parallelism. Being the basis for computations consuming a large fraction of cycles on the most capable computers, Sweep3D has been comprehensively modelled and executions studied on a wide range of platforms and scales (e.g., [8], [9], [10], [11], [12], [13]).

To investigate scaling behaviour of Sweep3D for a large range of scales, the benchmark input was configured with a fixed-size $32\times32\times512$ subgrid for each process: i.e., for an `NPE_I` by `NPE_J` grid of processes, total problem grid size is `IT_G`=32×`NPE_I`, `JT_G`=32×`NPE_J` and `KT`=512. Consistent with the benchmark and published studies, default values of `MK=10` and `MMI=3` were initially used for the blocking of k-planes and angles, respectively, which control the multitasking parallelism. 12 iterations were performed, with flux corrections (referred to as 'fixups') applied after 7 iterations. The code was built with IBM's mpxlf Fortran compiler and MPI library for Blue Gene, using the `-O3` optimization flag, and run on *Jugene* in virtual node (VN) mode using all four available cores per processor.

II. Base analysis of default Sweep3D execution configuration

Execution times reported for the timed Sweep3D kernel for a range of process counts are shown in Figure 1 (left graph, bold line with diamonds). From two minutes with 1,024 processes to over 8 minutes for 294,912 processes, a progressive slowdown is clear, which is consistent with that measured previously [12] and not uncommon when weak-scaling applications over such a large range.

To understand the execution performance behaviour, the Scalasca toolset (version 1.2) was employed. Sweep3D source routines were automatically instrumented using a common compiler capability, and the resulting objects linked with the Scalasca measurement library, such that events generated when entering and leaving user-program routines and operations in the MPI library could be captured and processed by the measurement library. Each Scalasca execution measurement and associated analysis is stored in a dedicated experiment archive directory. Elapsed times reported for the benchmark kernel of the uninstrumented version were within 5% of those when Scalasca measurements were made,



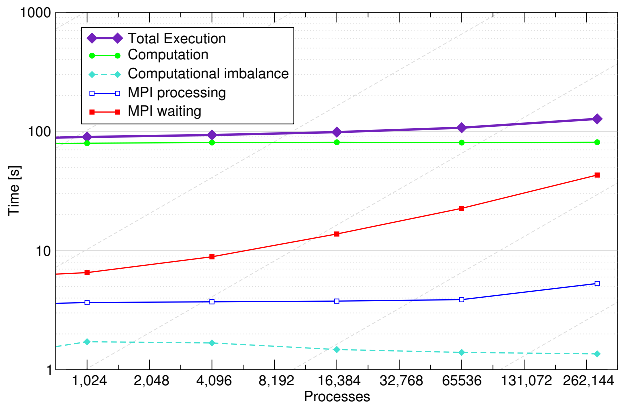
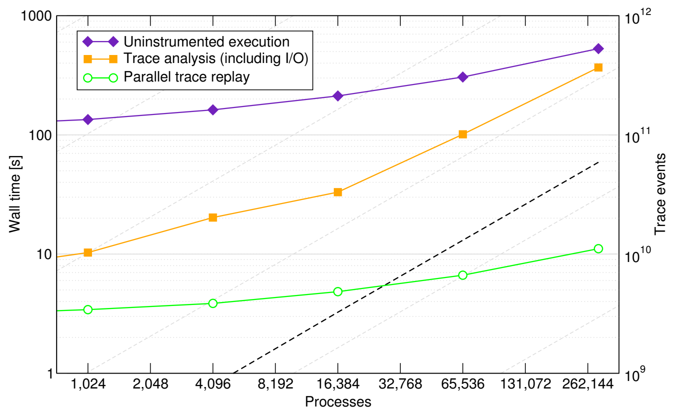


Figure 1. Scaling of Sweep3D execution time on BG/P for original MK=10 (left) and subsequently improved MK=1 (right) configurations of k-planes, with breakdown of computation and message-passing costs from Scalasca summary and trace analyses.



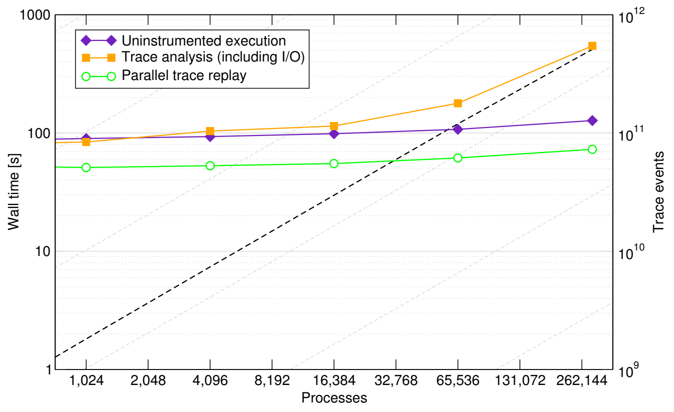


Figure 2. Scaling of Scalasca Sweep3D trace analysis times on BG/P, for original MK=10 (left) and subsequently improved MK=1 (right) configurations, with breakdown of parallel event replay time. (Total number of events captured in traces shown with dashed lines).

suggesting that instrumentation and measurement dilation were acceptable (and refinement was not needed).

A. *Runtime summarization*

An initial series of experiments were made using Scalasca runtime summarization to construct a callpath profile for each process during measurement, consisting of callpath visit count, execution time, and associated MPI statistics for the number and aggregate size of messages transferred. During measurement finalization, callpaths are unified and measurements collated into an XML report, that is subsequently post-processed into an analysis report consisting of 7 generic, 12 OpenMP-specific and 45 MPI-specific hierarchically organised metrics. Both report size and generation time increase linearly with the number of processes.

From the runtime summary profiles, it was found that the computation time (i.e., execution time excluding time in MPI operations) was 100 seconds, independent of scale, but the MPI communication time in the sweep kernel grew from some 30 to over 400 seconds. (MPI communication time is not shown in Figure 1, however, subsequent analysis will show that it is indistinguishable from MPI waiting time on the logarithmic scale.) Variation of time between ranks of around 10 seconds was also evident in marked patterns, where processes that had less computation time had an equivalently increased amount of communication time, due to the blocking point-to-point communication within each sweep and synchronizing `MPI_Allreduce` operations at the end of each sweep iteration. (Collective communication time itself was visibly concentrated on processes at the origin of the grid, and collective synchronization time is negligible since a `MPI_Barrier` is only employed at the very end of the timed computation and immediately following the synchronizing collective communication.)

An additional heuristic metric estimating *Computational imbalance* (shown as a dashed line in Figure 1), by calculating the absolute difference from the mean exclusive execution time of each callpath, could be readily employed to localize the imbalance to the main `sweep` routine, and validated by examining the distribution of execution times in that routine for each process. At larger scales, the imbalance is apparently less significant, therefore its further analysis was postponed while attention focussed on the message passing efficiency.

B. *Trace collection and analysis*

Even with all user routines instrumented and events for all MPI operations, scoring of the summary profile analysis report determined that the size of trace buffer required for each process was only 2.75MB. Since this is less than the Scalasca default value of 10MB, and the majority of this space was for MPI events, trace collection and analysis required no special configuration of trace buffers or filters. Storing trace event data in a separate file for each process, Scalasca trace analysis proceeds automatically after measurement is complete using the same configuration of processes to replay the traced events in a scalable fashion. Figure 2 shows that trace analysis times (squares) remain modest, even though total sizes of traces increase linearly with the number of processes to 790GB for 59e9 traced events (dashed line). Although trace files are read in parallel, the linearly increasing time for generation of the final analysis report (identical to the summary report augmented with 20 trace-specific MPI metrics) dominates at larger scales. By far the most significant hindrance to the scalability of such trace-based analysis — and also applications which use similar techniques for intermediate checkpoints and final results — is the creation of one file per process, which grew to take over 86 minutes for 294,912 files, apparently due to GPFS filesystem metadata-server contention.

From Scalasca automated trace analysis examining event patterns and quantifying their corresponding cost, MPI communication time can be split into basic message processing and a variety of performance properties indicating waiting time when message-passing operations are blocked from proceeding (see Figure 3 for examples). Figure 1 shows that while basic message processing time (open squares) remains negligible and fairly constant at around one second, MPI communication time is dominated by increasingly onerous waiting time (filled squares) that governs the performance of Sweep3D at larger scales. Most waiting time is found to be *Late Sender* situations, where a blocking receive is initiated earlier than the associated send, with further waiting time for *Wait at N x N* in the `MPI_Allreduce` operations for processes that initiate the collective operation in advance of the last participant.

C. *Auxilliary investigation of process mappings*

Since similar (though less extreme) behaviour was observed with smaller process configurations, experiments were repeated specifying alternative mappings of processes onto the BG/P physical torus hardware. The default XYZT mapping was found to be statistically as good as permutations of XYZ, while TXYZ (and permutations) which map consecutive ranks onto the same processor cores degraded performance by some 2%. In comparison, optimal mappings have been reported to be able to improve Sweep3D performance by 4% on Blue Gene/L [12]. (It would be interesting to investigate the performance difference using the hardware counters available on BG/P, however, meaningful measurements are currently only possible in SMP mode.) Scalasca experiments using a Cray XT system with 4 cores and processes per compute node had comparable MPI costs and patterns of imbalance, indicating that the communication network and mapping of processes are not pertinent to the communication overhead and imbalance.

III. Detailed analysis of improved Sweep3D execution configuration

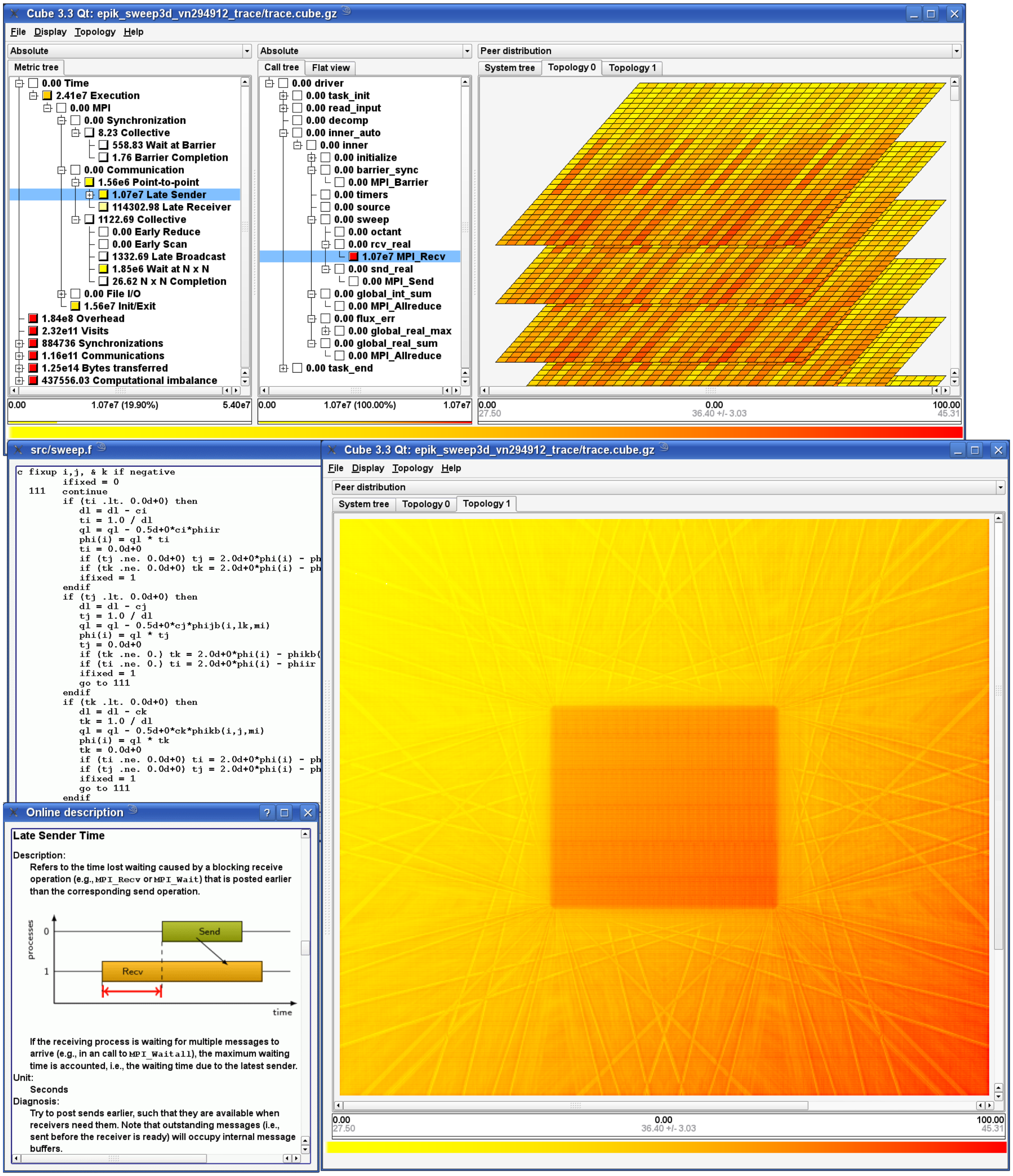
Blocking of k-planes and angles in Sweep3D can significantly change the computation/communication ratio, with a trade-off between fewer communication steps with larger message sizes and better parallel efficiency from more rapid succession of wavefronts [8], [10]. From trials varying the numbers of k-planes in a block of grid points (`MK`) and angles processed together (`MMI`), using single k-planes (`MK=1`) was found to be optimal, and more than 4 times faster than the default (`MK=10`) with 294,912 processes on BG/P. (The number of angles didn't need to be adjusted.)

A. *Revised summary and trace analyses*

Having determined from new Scalasca summary experiments that MPI communication costs are substantially reduced on BG/P with the additional pipelining possible in the `MK=1` configuration, general performance and scaling behaviour were now much better as shown in Figure 1 (right). Total execution time is reduced for all process configurations, especially at the largest scales. (Measurement dilation was even reduced to only 3%.)

Scalasca tracing experiments were also repeated with the new configuration to investigate the performance change. Recently incorporated support for the SIONlib library [14] was employed, such that one multi-file was created by each BG/P I/O node (i.e., 576 files at full scale) for the traces from 512 processes, reducing the time for creation of the experiment archive directory and trace files from 86 minutes (for individual files) down to 10 minutes. With ten times more messages due to the smaller computational blocks, 27MB trace buffers were required, and traces are correspondingly larger and slower to analyse, as seen in Figure 2, however, parallel trace replay time (circles) closely parallels Sweep3D execution time. For 294,912 processes, 510e9 events were recorded and total trace size was 7.6TB. Serial writing of the analysis report was also improved 8-fold by dumping gathered metric severity values in binary format (while retaining XML for the metadata header).

Sweep3D computation time is found to be 20% less than previously at a constant 80 seconds. MPI processing times increased from 1 to 4 seconds due to the additional messages transferred, however, remain relatively insignificant. Most notable is the reduction in MPI waiting times, which although still increasing with scale, no longer dominate overall performance as they did before.



```
0.00 Time
  2.41e7 Execution
    0.00 MPI
      0.00 Synchronization
        8.23 Collective
          558.83 Wait at Barrier
          1.76 Barrier Completion
      0.00 Communication
        1.56e6 Point-to-point
          1.07e7 Late Sender
          114302.98 Late Receiver
        1122.69 Collective
          0.00 Early Reduce
          0.00 Early Scan
          1332.69 Late Broadcast
          1.85e6 Wait at N x N
          26.62 N x N Completion
      0.00 File I/O
    1.56e7 Init/Exit
1.84e8 Overhead
2.32e11 Visits
884736 Synchronizations
1.16e11 Communications
1.25e14 Bytes transferred
437556.03 Computational imbalance

0.00        1.07e7 (19.90%)        5.40e7
```

```
0.00 driver
  0.00 task_init
  0.00 read_input
  0.00 decomp
  0.00 inner_auto
    0.00 inner
      0.00 initialize
      0.00 barrier_sync
        0.00 MPI_Barrier
      0.00 timers
      0.00 source
      0.00 sweep
        0.00 octant
        0.00 rcv_real
          1.07e7 MPI_Recv
        0.00 snd_real
          0.00 MPI_Send
      0.00 global_int_sum
        0.00 MPI_Allreduce
      0.00 flux_err
        0.00 global_real_max
      0.00 global_real_sum
        0.00 MPI_Allreduce
  0.00 task_end

0.00        1.07e7 (100.00%)        1.07e7
```

```
0.00                  0.00                  100.00
27.50            36.40 +/- 3.03              45.31
```

```
c fixup i,j, & k if negative
      ifixed = 0
 111  continue
      if (ti .lt. 0.0d+0) then
         dl = dl - ci
         ti = 1.0 / dl
         ql = ql - 0.5d+0*ci*phiir
         phi(i) = ql * ti
         ti = 0.0d+0
         if (tj .ne. 0.0d+0) tj = 2.0d+0*phi(i) - ph
         if (tk .ne. 0.0d+0) tk = 2.0d+0*phi(i) - ph
         ifixed = 1
      endif
      if (tj .lt. 0.0d+0) then
         dl = dl - cj
         tj = 1.0 / dl
         ql = ql - 0.5d+0*cj*phijb(i,lk,mi)
         phi(i) = ql * tj
         tj = 0.0d+0
         if (tk .ne. 0.) tk = 2.0d+0*phi(i) - phikb(
         if (ti .ne. 0.) ti = 2.0d+0*phi(i) - phiir
         ifixed = 1
         go to 111
      endif
      if (tk .lt. 0.0d+0) then
         dl = dl - ck
         tk = 1.0 / dl
         ql = ql - 0.5d+0*ck*phikb(i,j,mi)
         phi(i) = ql * tk
         tk = 0.0d+0
         if (ti .ne. 0.0d+0) ti = 2.0d+0*phi(i) - ph
         if (tj .ne. 0.0d+0) tj = 2.0d+0*phi(i) - ph
         ifixed = 1
         go to 111
      endif
```

Late Sender Time

Description:
Refers to the time lost waiting caused by a blocking receive operation (e.g., `MPI_Recv` or `MPI_Wait`) that is posted earlier than the corresponding send operation.

If the receiving process is waiting for multiple messages to arrive (e.g., in an call to `MPI_Waitall`), the maximum waiting time is accounted, i.e., the waiting time due to the latest sender.

Unit:
Seconds

Diagnosis:
Try to post sends earlier, such that they are available when receivers need them. Note that outstanding messages (i.e., sent before the receiver is ready) will occupy internal message buffers.

```
0.00                  0.00                  100.00
27.50            36.40 +/- 3.03              45.31
```

Figure 3. Scalasca analysis report explorer presentation of Sweep3D execution performance with 294,912 MPI processes on Blue Gene/P. The physical three-dimensional torus topology (upper right) and 576×512 application virtual topology (lower right, grid origin in NW corner) show the distribution of values per process, for the *Late Sender* waiting time metric selected from the metric hierarchy (upper left) for the `MPI_Recv` callpath. Processes in the topology displays and the boxes next to nodes in the trees are colour-coded by metric value according to the colour scale at the bottom of the window.

The computational imbalance that appeared to be associated with the communication waiting time is still found in the new configuration, with amplified significance. As before, the imbalance is in the `sweep` routine, however, it is now a much larger proportion of the reduced computation time, and has more of an effect on the associated communication waiting time. Figure 3 shows the presentation of the Sweep3D execution performance with 294,912 MPI processes by the Scalasca analysis report explorer. The program calltree in the middle panel has a similarly tree-structured set of performance metrics in the left panel, and on the right a depiction of the MPI processes according to the BG/P physical hardware topology. The profile of the selected *Late Sender* metric from the *Time* metric hierarchy shows all of it is found in `MPI_Recv` calls in the `sweep` routine, and the distribution of values by process. Switching to a two-dimensional 576×512 grid corresponding to the application virtual topology (lower right) reveals the distribution of *Late Sender* time with several pronounced characteristics: a background of waiting times progressively increasing from NW to SE corners, a central rectangular block with higher waiting time, and an intricate pattern of sharp oblique lines radiating from the central block to the edges.

B. Refined instrumentation and analyses

To isolate the origin of the imbalance, the Sweep3D source code was manually annotated with Scalasca instrumentation macros. Starting with the key 625-line `sweep` flow routine, the loop over the eight octants was annotated to define a distinct region in the callpath when processing each octant. It was found that computation times didn't vary much by octant, however, there was a sizable variation in communication time between octants (which will be re-examined later in more detail).

With further annotation of the execution phases within octant processing, the imbalance was isolated to the i-line section where corrective 'fixups' for negative fluxes are recursively applied in the i, j, k directions (shown in Scalasca source browser window in Figure 3), as typically identified as a hotspot by sampling-based profilers (e.g., [13]). Finer annotation of the conditional fixup block for each direction determined that i and j corrections are applied with roughly the same frequency, and somewhat more often than k corrections. In each case, there is a pronounced distribution pattern, varying from a relatively small number of fixes in an interior rectangular block with much higher numbers on oblique lines reaching to the border of the domain (matching that visible in Figure 3). The aggregate computation time for applying these fixes is directly proportional to the number of fixes applied. Since this computation is done between receiving inflows and sending outflows for each block of k-planes, delays sending outflows on processes applying more flux corrections result in additional waiting time in receives for inflows on neighbours.

Since the input configuration for Sweep3D specifies that flux fixups are only applied after the seventh iteration, the major solver iteration 'loop' in the `inner` routine was annotated: this 'loop' with increasing values of `its` is implicitly defined by a `continue` statement and a guarded `goto` statement, within which region entry and exit annotations were incorporated, each time defining a new region labeled with the corresponding value of `its`. Each of the 12 executions of this region was then distinguished in the resulting callpath analysis, visible in the middle panel of the screenshot at the top of Figure 4. Charts of the execution time for each iteration can also be produced, with a breakdown of the MPI processing and waiting times, such as shown in Figure 5. While the initial seven iterations have very similar performance characteristics, including minimal imbalance in computation or communication, the eighth iteration is markedly more expensive with significant imbalance. Subsequent iterations are not quite so bad, however, they still have significant imbalance and waiting times, with a pattern that spreads from the central rectangular block along oblique angles out to the edges visible in the sequence of views of the process computation time distribution in Figure 4. (A colour scale for times from 5 to 10 seconds is used to enhance contrast: the initial 6 iterations are indistinguishable from iteration 7, and the final 2 iterations are very similar to iteration 10.)

Separating the analysis of the computationally-balanced non-fixup iterations from that of the iterations with computationally-imbalanced fixup calculations, helps distinguish the general efficiency of the communication sweeps from additional inefficiencies arising from the computational imbalance. In this case, octant instrumentation is combined with instrumentation that selects between fixup and non-fixup iterations, producing a profile as shown in Figure 6. Here the distribution of *Late Sender* waiting time is a complement to the distribution of pure computation time arising from the fixup calculations seen in Figure 4. Communication time for even-numbered octants is negligible for the non-fixup iterations (which are also well balanced), and while octants 1, 3, and 7 have comparable communication times, octant 5 generally requires twice as long: this octant is where the sweep pipeline must drain before the reverse sweep can be initiated, with corresponding waiting time. The distribution of *Late Sender* waiting time in non-fixup iterations for pairs of octants shown in Figure 6 illustrates the impact of the sweeps. In octants 1+2, waiting times are greatest in the NW and progressively diminish towards the SE. For octants 5+6, the waiting times are larger due to the sweep reversal, and the progression is from NE to SW. Octants 3+4 and 7+8 combine sweeps from both SW to NE and SE to NW resulting in progressively decreasing amounts of waiting time from south to north. Each octant in fixup iterations has more than twice as much aggregate *Late Sender* waiting time, with a distribution that clearly

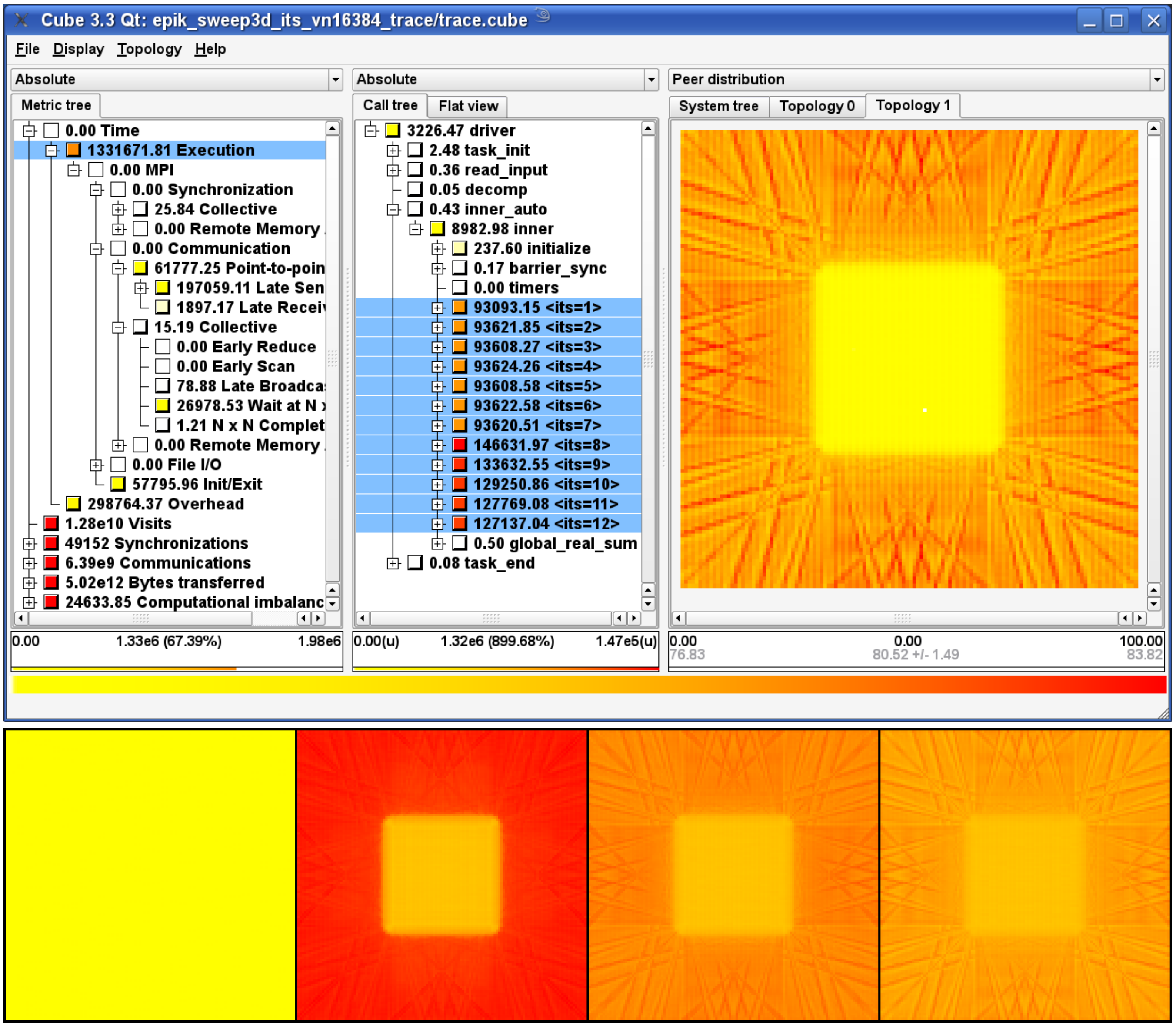
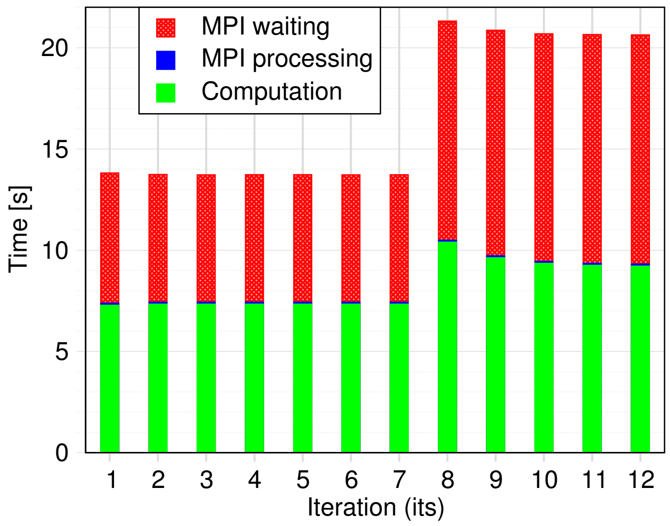

```
Cube 3.3 Qt: epik_sweep3d_its_vn16384_trace/trace.cube
File  Display  Topology  Help

Absolute                         Absolute                        Peer distribution
Metric tree                      Call tree   Flat view           System tree   Topology 0   Topology 1

0.00 Time                        3226.47 driver
  1331671.81 Execution             2.48 task_init
    0.00 MPI                       0.36 read_input
      0.00 Synchronization         0.05 decomp
        25.84 Collective           0.43 inner_auto
        0.00 Remote Memory           8982.98 inner
      0.00 Communication               237.60 initialize
        61777.25 Point-to-poin         0.17 barrier_sync
          197059.11 Late Sen           0.00 timers
          1897.17 Late Recei           93093.15 <its=1>
        15.19 Collective               93621.85 <its=2>
          0.00 Early Reduce            93608.27 <its=3>
          0.00 Early Scan              93624.26 <its=4>
          78.88 Late Broadca           93608.58 <its=5>
          26978.53 Wait at N x         93622.58 <its=6>
          1.21 N x N Complet           93620.51 <its=7>
        0.00 Remote Memory             146631.97 <its=8>
      0.00 File I/O                    133632.55 <its=9>
      57795.96 Init/Exit               129250.86 <its=10>
    298764.37 Overhead                 127769.08 <its=11>
  1.28e10 Visits                       127137.04 <its=12>
  49152 Synchronizations               0.50 global_real_sum
  6.39e9 Communications            0.08 task_end
  5.02e12 Bytes transferred
  24633.85 Computational imbalanc

0.00   1.33e6 (67.39%)   1.98e6  0.00(u)  1.32e6 (899.68%)  1.47e5(u)  0.00       0.00            100.00
                                                                       76.83  80.52 +/- 1.49     83.82
```

Figure 4. Sweep3D computation *Execution* time variation by iteration (top) and 16,384-process distribution evolution for iterations 7 to 10 (bottom).

superimposes the underlying sweep with the additional computational imbalance.

IV. Conclusion

The ubiquitous Sweep3D benchmark code has good scalability to very high numbers of processes, however, careful evaluation of coupled input parameters is required to ensure that waiting times for MPI communication do not grow to dominate execution performance. Although Sweep3D has been comprehensively studied and modelled, providing valuable insight into expected performance, actual execution at extreme scales can differ appreciably due to easily overlooked factors that introduce substantial imbalance and additional waiting times. Key execution performance characteristics of Sweep3D were revealed by Scalasca runtime summarization and automated event trace analyses, and refined employing source code annotations inserted for major iteration loops and code sections to direct instrumentation and analysis. In on-going research we are investigating automatic determination and combining of iterations with similar performance profiles [15], and analysing traces for the root causes of wait states to improve attribution of performance problems [16]. Tools for measuring and analysing application execution performance also need to be highly scalable, as demonstrated by the Scalasca toolset with 294,912 Sweep3D processes on Blue Gene/P, where multiple



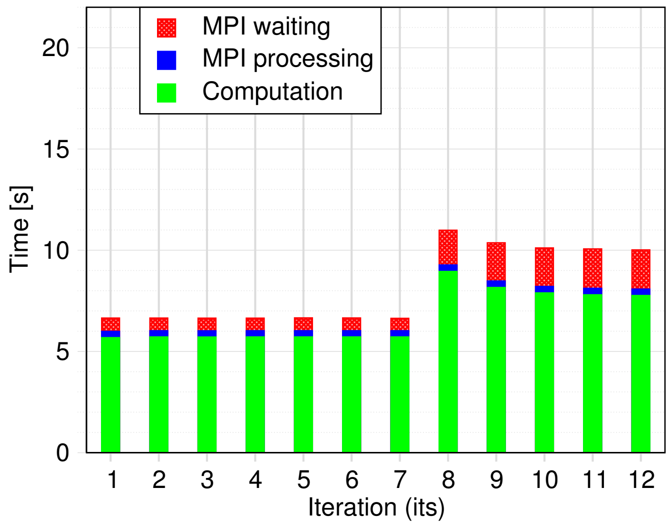


Figure 5. Sweep3D iteration execution time breakdown with 16,384 processes on BG/P for MK=10 (left) and MK=1 (right) k-plane blocking factors.

techniques for effective data reduction and management are employed and application-oriented graphical presentation facilitated insight into load-balance problems that only become critical at larger scales.

Acknowledgements: We would like to thank our JSC colleagues for their assistance in performing this work, and the reviewers for their insightful suggestions.

References

[1] Jülich Supercomputing Centre, "Scalasca toolset for scalable performance analysis of large-scale parallel applications," http://www.scalasca.org/.

[2] M. Geimer, F. Wolf, B. J. N. Wylie, E. Ábrahám, D. Becker, and B. Mohr, "The Scalasca performance toolset architecture," *Concurrency and Computation: Practice and Experience*, DOI: 10.1002/cpe.1556 (to appear).

[3] F. Wolf, B. J. N. Wylie, E. Ábrahám, D. Becker, W. Frings, K. Fürlinger, M. Geimer, M.-A. Hermanns, B. Mohr, S. Moore, M. Pfeifer, and Z. Szebenyi, "Usage of the SCALASCA toolset for scalable performance analysis of large-scale parallel applications," in *Proc. 2nd HLRS Parallel Tools Workshop (Stuttgart, Germany)*. Springer, Jul. 2008, pp. 157–167, ISBN 978-3-540-68561-6.

[4] B. J. N. Wylie, M. Geimer, and F. Wolf, "Performance measurement and analysis of large-scale parallel applications on leadership computing systems," *Journal of Scientific Programming*, vol. 16, no. 2–3, pp. 167–181, 2008.

[5] IBM Blue Gene team, "Overview of the IBM Blue Gene/P project," *IBM Journal of Research and Development*, vol. 52, no. 1/2, pp. 199–220, Jan. 2008.

[6] Los Alamos National Laboratory, "ASCI SWEEP3D v2.2b: 3-dimensional discrete ordinates neutron transport benchmark," http://wwwc3.lanl.gov/pal/software/sweep3d/, 1995.

[7] A. Hoisie, O. M. Lubeck, and H. J. Wasserman, "Performance analysis of wavefront algorithms on very-large scale distributed systems," in *Proc. Workshop on Wide Area Networks and High Performance Computing*, ser. Lecture Notes in Control and Information Sciences, vol. 249. Springer, 1999, pp. 171–187.

[8] A. Hoisie, O. Lubeck, and H. Wasserman, "Performance and scalability analysis of Teraflop-scale parallel architectures using multidimensional wavefront applications," *Int'l Journal of High Performance Computing Applications*, vol. 14, pp. 330–346, 2000.

[9] D. H. Ahn and J. S. Vetter, "Scalable analysis techniques for microprocessor performance counter metrics," in *Proc. ACM/IEEE SC 2002 conference (Baltimore, MD, USA)*. IEEE Computer Society Press, Nov. 2002.

[10] K. Davis, A. Hoisie, G. Johnson, D. J. Kerbyson, M. Lang, S. Pakin, and F. Petrini, "A performance and scalability analysis of the Blue Gene/L architecture," in *Proc. ACM/IEEE SC 2004 conference (Pittsburgh, PA, USA)*. IEEE Computer Society Press, Nov. 2004.

[11] M. M. Mathis and D. J. Kerbyson, "A general performance model of structured and unstructured mesh particle transport computations," *Journal of Supercomputing*, vol. 34, no. 2, pp. 181–199, Nov. 2005.

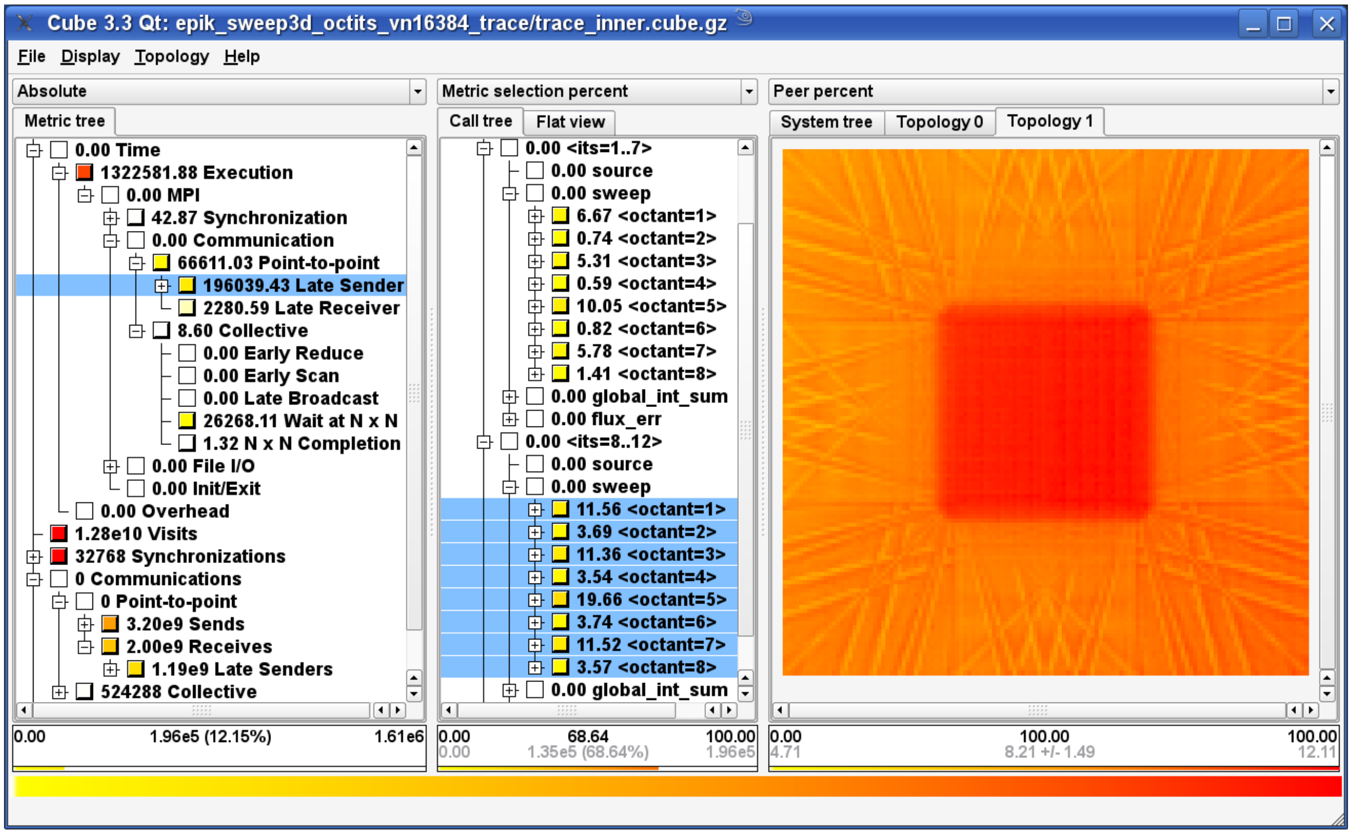
[12] A. Hoisie, G. Johnson, D. J. Kerbyson, M. Lang, and S. Pakin, "A performance comparison through benchmarking and modeling of three leading supercomputers: Blue Gene/L, Red Storm, and Purple," in *Proc. ACM/IEEE SC 2006 conference (Tampa, FL, USA)*. IEEE Computer Society Press, Nov. 2006.

[13] A. Bordelon, "Developing a scalable, extensible parallel performance analysis toolkit," Master's thesis, Rice University, Houston, TX, USA, Apr. 2007.

[14] W. Frings, F. Wolf, and V. Petkov, "Scalable massively parallel I/O to task-local files," in *Proc. ACM/IEEE SC 2009 conference (Portland, OR, USA)*. IEEE Computer Society Press, Nov. 2009.

[15] Z. Szebenyi, F. Wolf, and B. J. N. Wylie, "Space-efficient time-series call-path profiling of parallel applications," in *Proc. ACM/IEEE SC 2009 conference (Portland, OR, USA)*. IEEE Computer Society Press, Nov. 2009.

[16] D. Böhme, M. Geimer, M.-A. Hermanns, and F. Wolf, "Identifying the root causes of wait states in large-scale parallel applications," Aachen Institute for Advanced Study in Computational Engineering Science, RWTH Aachen University, Aachen, Germany, Tech. Rep. AICES-2010-1, Jan. 2010.



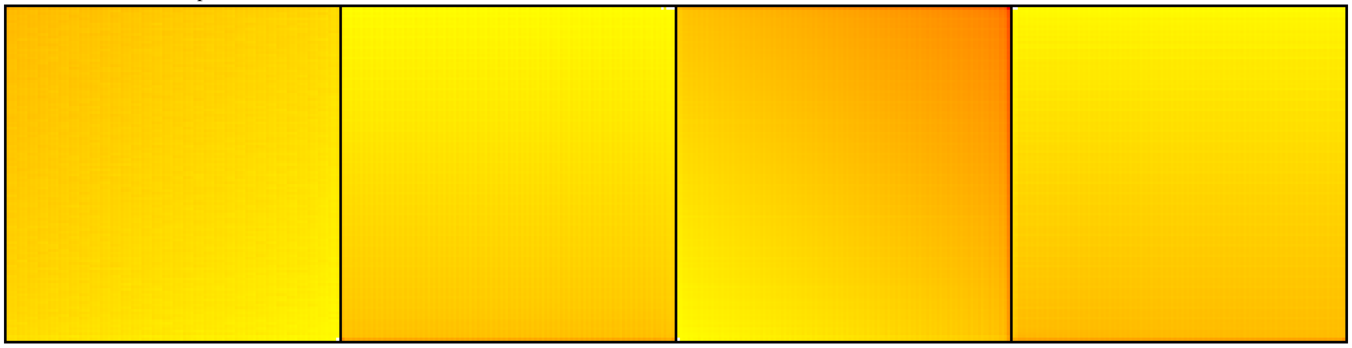
Cube 3.3 Qt: epik_sweep3d_octits_vn16384_trace/trace_inner.cube.gz

```
0.00 Time
  1322581.88 Execution
    0.00 MPI
      42.87 Synchronization
      0.00 Communication
        66611.03 Point-to-point
          196039.43 Late Sender
          2280.59 Late Receiver
        8.60 Collective
          0.00 Early Reduce
          0.00 Early Scan
          0.00 Late Broadcast
          26268.11 Wait at N x N
          1.32 N x N Completion
      0.00 File I/O
      0.00 Init/Exit
  0.00 Overhead
1.28e10 Visits
32768 Synchronizations
0 Communications
  0 Point-to-point
    3.20e9 Sends
    2.00e9 Receives
      1.19e9 Late Senders
  524288 Collective
```

```
0.00 <its=1..7>
  0.00 source
  0.00 sweep
    6.67 <octant=1>
    0.74 <octant=2>
    5.31 <octant=3>
    0.59 <octant=4>
    10.05 <octant=5>
    0.82 <octant=6>
    5.78 <octant=7>
    1.41 <octant=8>
  0.00 global_int_sum
  0.00 flux_err
0.00 <its=8..12>
  0.00 source
  0.00 sweep
    11.56 <octant=1>
    3.69 <octant=2>
    11.36 <octant=3>
    3.54 <octant=4>
    19.66 <octant=5>
    3.74 <octant=6>
    11.52 <octant=7>
    3.57 <octant=8>
  0.00 global_int_sum
```

```
0.00          1.96e5 (12.15%)        1.61e6
0.00          68.64                  100.00
0.00          1.35e5 (68.64%)        1.96e5
0.00          100.00                 100.00
4.71          8.21 +/- 1.49          12.11
```

Iterations without fixups:



Iterations with fixups:

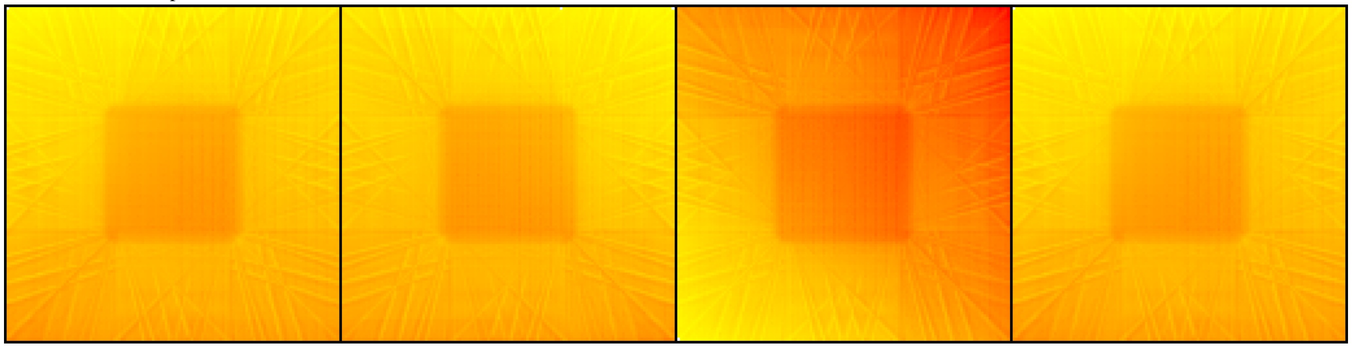


Figure 6. Sweep3D MPI communication *Late Sender* time variation by sweep octant for initial 7 non-fixup and subsequent 5 fixup iterations (top) and 16,384-process waiting time distributions for the computationally balanced non-fixup and imbalanced fixup octant pairs 1+2, 3+4, 5+6, 7+8 (bottom).