Parallel Processing Letters
Vol. 20, No. 4 (2010) 397–414

LARGE-SCALE PERFORMANCE ANALYSIS OF SWEEP3D WITH THE SCALASCA TOOLSET

BRIAN J. N. WYLIE, MARKUS GEIMER, BERND MOHR
Jülich Supercomputing Centre, Forschungszentrum Jülich, 52425 Jülich, Germany

and

DAVID BÖHME, ZOLTÁN SZEBENYI
Jülich Supercomputing Centre, Forschungszentrum Jülich, 52425 Jülich, Germany
RWTH Aachen University, 52056 Aachen, Germany

and

FELIX WOLF
German Research School for Simulation Sciences, 52062 Aachen, Germany
Jülich Supercomputing Centre, Forschungszentrum Jülich, 52425 Jülich, Germany
RWTH Aachen University, 52056 Aachen, Germany

Received June 2010
Revised August 2010
Communicated by Guest Editors

ABSTRACT

Cray XT and IBM Blue Gene systems present current alternative approaches to constructing leadership computer systems relying on applications being able to exploit very large configurations of processor cores, and associated analysis tools must also scale commensurately to isolate and quantify performance issues that manifest at the largest scales. In studying the scalability of the Scalasca performance analysis toolset to several hundred thousand MPI processes on XT5 and BG/P systems, 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, in a portable and straightforward to use toolset, Scalasca was thereby able to demonstrate performance measurements and analyses with 294,912 processes.

Keywords: parallel performance measurement & analysis; scalability

1. 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 two of the largest leadership computers each with more than 200 thousand cores — the *Jaguar* Cray XT5 system [5] at Oak Ridge National Laboratory and *Jugene* IBM Blue Gene/P system [6] at Jülich Supercomputing Centre — we chose to study the highly-scalable compact application Sweep3D.

After introducing the Sweep3D test code and the Scalasca toolset, we investigate the weak-scaling behaviour of Sweep3D on XT5 and BG/P via a series of Scalasca measurement and analysis experiments on each system. As well as describing how Scalasca was applied, and comparing execution characteristics, scaling issues are reviewed and particular attention is given to executions with very large configurations of MPI processes. While much can be learned from experiments at relatively small and intermediate scales, it is instructive to examine how application execution behaviour changes with scale, as well as verifying that Scalasca continues to be usable. For this purpose, some of the largest experiments performed on both systems are then inspected in detail. This is followed by a more refined instrumentation and analysis of Sweep3D execution at more modest scale for additional insight into the origin of the performance issues that dominate at scale on both platforms.

2. SWEEP3D

The ASCI Sweep3D benchmark code [7, 8] 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 modeled and executions studied on a wide range of platforms and scales (e.g., [9, 10, 11, 12, 13, 14]).

The code was built on XT5 and BG/P with the vendor-provided MPI libraries and Fortran compilers (PGI and IBM XL respectively), using the `-O3` optimization flag, and run using all available cores per processor (i.e., 6 for XT5 and 4 for BG/P) and an MPI process on each core.

To investigate scaling behaviour of Sweep3D for a large range of scales, the benchmark input was configured with a fixed-size $32 \times 32 \times 512$ subgrid for each process: i.e., for an `NPE_I` by `NPE_J` grid of processes, the 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, 12 iterations were performed, with flux corrections (referred to as 'fixups') applied after 7 iterations.

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. These parameters 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 [9, 11]. 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 on BG/P. (The number of angles didn't need to be adjusted.) In contract, no improvement over the default configuration was found on the XT5.

3. Scalasca

The open-source Scalasca toolset supports measurement and analysis of MPI applications written in C, C++ and Fortran on a wide range of current HPC platforms [3, 4]. Hybrid codes making use of basic OpenMP features in addition to message passing are also supported. Figure 1 shows the Scalasca workflow for instrumentation, measurement, analysis and presentation.

Before performance data can be collected, the target application must be instrumented and linked to the measurement library. The instrumenter for this purpose is used as a prefix to the usual compile and link commands, offering a variety of manual and automatic instrumentation options. MPI operations are captured simply via re-linking, whereas a source preprocessor is used to instrument OpenMP parallel regions. Often compilers can be directed to automatically instrument the entry and exits of user-level source routines, or the PDToolkit source-code instrumenter can be used for more selective instrumentation of routines. Finally, programmers can manually add custom instrumentation annotations into the source code for impor-

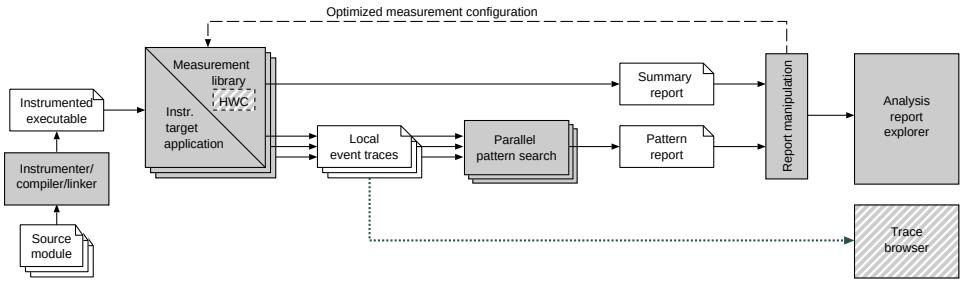


Fig. 1. Schematic overview of Scalasca instrumentation, measurement, analysis and presentation.

tant regions via macros or pragmas which are ignored when instrumentation is not activated.

The Scalasca measurement and analysis nexus configures and manages collection of execution performance experiments, which is similarly used as a prefix to the parallel execution launch command of the instrumented application executable (i.e., `aprun` on XT5 and `mpirun` on BG/P) and results in the generation of a unique experiment archive directory containing measurement and analysis artifacts, including log files and configuration information.

Users can choose between generating a summary analysis report ('profile') with aggregate performance metrics for each function callpath and/or generating event traces recording runtime events from which a profile or time-line visualization can later be produced. Summarization is particularly useful to obtain an overview of the performance behaviour and for local metrics such as those derived from hardware counters. Since measurement overheads can be prohibitive for small routines that are executed often and traces tend to rapidly become very large, optimizing the instrumentation and measurement configuration based on the summary report is usually recommended. When tracing is enabled, each process generates a trace containing records for its process-local events: by default separate files are created for each MPI rank, or SIONlib can be used to improve file handling by transparently mapping task-local files into a smaller number of physical files [15]. After program termination (and with the same partition of processors), the Scalasca nexus automatically loads the trace files into main memory and analyzes them in parallel using as many processes as have been used for the target application itself. During trace analysis, Scalasca searches for wait states and related performance properties, classifies detected instances by category, and quantifies their significance. The result is a wait-state report similar in structure to the summary report but enriched with higher-level communication and synchronization inefficiency metrics.

Both summary and wait-state reports contain performance metrics for every measured function callpath and process/thread which can be interactively examined in the provided analysis report explorer. Prior to initial presentation, raw measurement reports are processed to derive additional metrics and structure the resulting set of metrics in hierarchies. Additional processing to combine reports or extract sections produces new reports with the same format. Scalasca event traces may also be examined directly (or after conversion if necessary) by various third-party trace visualization and analysis tools.

4. Base Analysis of SWEEP3D Executions

Execution times reported for the timed Sweep3D kernel for a range of process counts are shown in Figure 2 (left column of graphs, bold lines with diamonds). Sweep3D was run with up to 196,608 processes on the Cray XT5 and 294,912 processes on IBM BG/P, taking 121 and 505 seconds respectively, both using the default configuration of 10 k-plane blocks (`MK=10`). Progressive slowdown with increasing scale is clear

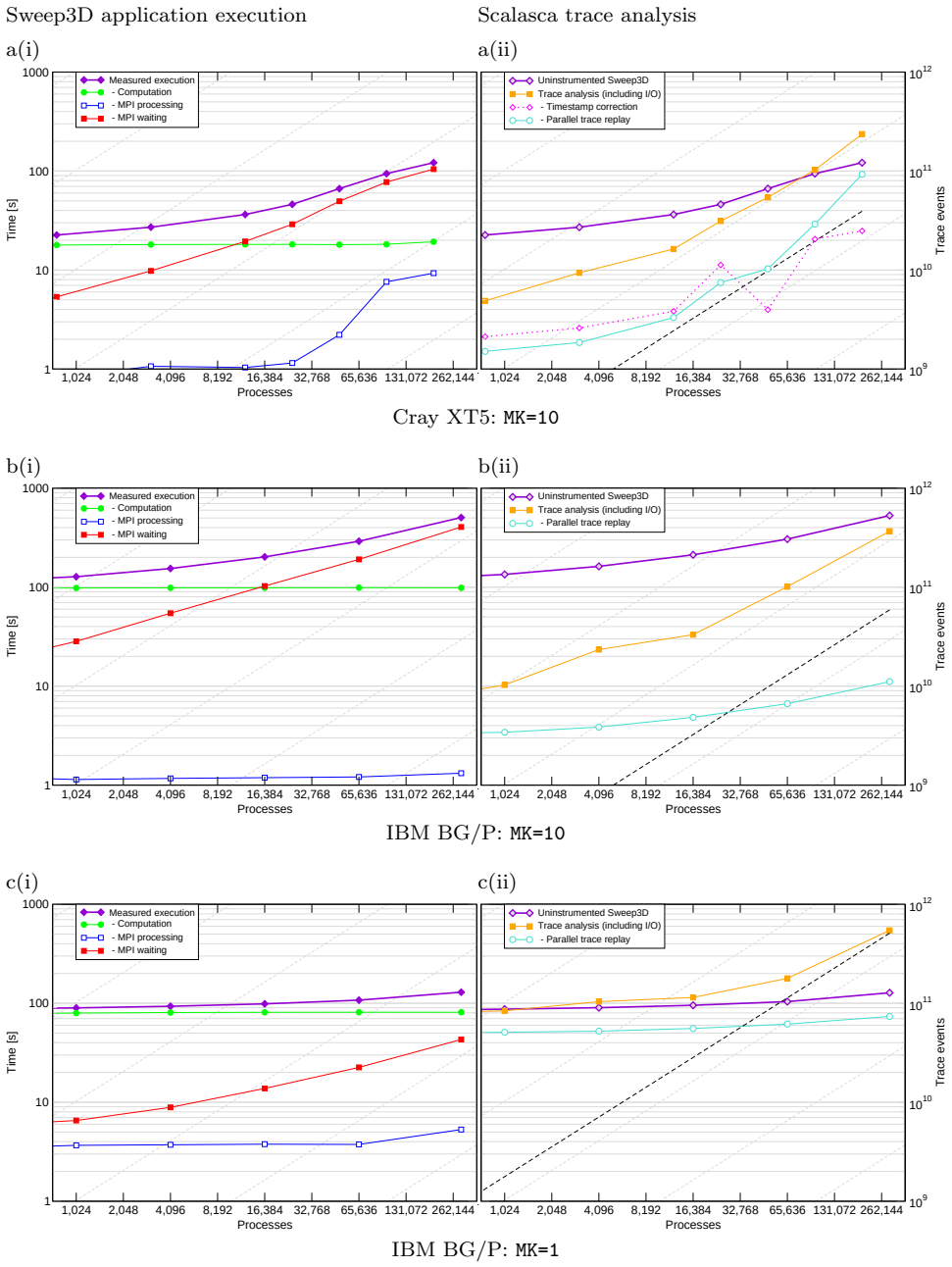


Fig. 2. Scaling of Sweep3D execution time and Scalasca trace analysis time on Cray XT5 (top row) and IBM BG/P (middle and bottom rows, original and improved configurations). Sweep3D measured execution time (left column) is separated into computation and message-passing costs from Scalasca summary and trace analyses. Scalasca trace analysis time (right column) includes a breakdown of times for timestamp correction (not required on BG/P) and parallel event replay. (Dashed black line is the total number of trace event records with the scale on the right side.)

which is consistent with that measured previously [13] and not uncommon when weak-scaling applications over such a large range. A further series of measurements taken on the IBM BG/P is also shown, using single k-plane blocks (`MK=1`) which were found to substantially improve performance at larger scale, reducing execution time to 129 seconds with 294,912 processes: while this is a dramatic 3.9 times faster, there is also a 20% improvement with only 1,024 processes.

With the default `MK=10` configurations, execution time grows by a factor of 5.4 for 256 times as many processes on the XT5, and a factor of 4.0 for 288 times as many processes on BG/P. Scalability on BG/P is therefore somewhat superior with comparable execution configurations, though the base performance is considerably poorer, and with the `MK=1` configuration both basic performance and scalability are notably improved.

To understand the execution performance behaviour, the Scalasca toolset was employed on both XT5 and BG/P. Sweep3D source routines were automatically instrumented using a common capability of the PGI and IBM XL compilers, 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. Elapsed times reported for the benchmark kernel of the uninstrumented version were within 5% of those when Scalasca measurements were made, suggesting that instrumentation and measurement dilation were acceptable (and refinement was not needed).

4.1. *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 independent of scale, but the MPI communication time in the sweep kernel rapidly grows to dominate total execution time. (MPI communication time is not shown in Figure 2, however, subsequent analysis will show that it is indistinguishable from MPI waiting time on the logarithmic scale.) The 18 seconds of computation time on XT5 is exceeded by MPI communication time for 12,288 processes, and the 100 seconds of computation time on BG/P exceeded by MPI communication time for 16,384 processes for `MK=10`. With `MK=1` on BG/P, computation time is only 80 seconds and MPI communication time is greatly reduced, though still growing with scale.

Variation of time between ranks of around 10% was also evident in marked pat-

terns, 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 concentrated on processes at the origin of the application 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.)

4.2. *Trace Collection and Analysis*

Even with all user routines instrumented and events for all MPI operations, scoring of the `MK=10` summary profile analysis reports determined that the size of trace buffer required for each process was only 2.75 MB. Since this is less than the Scalasca default value of 10 MB, and the majority of this space was for MPI events, trace collection and analysis required no special configuration of trace buffers or filters. As the `MK=1` configuration results in ten times as many messages, trace buffers of 30 MB per process were specified via the appropriate experiment configuration variable for those experiments.

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. The right column of Figure 2 shows that trace analysis times (squares) remain modest, even though total sizes of traces increase linearly with the number of processes (dashed line). 59e9 traced events for the BG/P `MK=10` execution occupied 790 GB, while the `MK=1` configuration produced a total of 510e9 events and a trace of 7.6 TB.

Characteristics of the largest Scalasca trace measurements and analyses of Sweep3D on Cray XT and IBM BG/P are summarized in the following table.

Table 1. Scalasca Sweep3D trace experiment statistics.

	Jaguar	*Jugene*	
Architecture	Cray XT5	IBM Blue Gene/P	
Processes	196,608	294,912	294,912
Sweep3D size of k-blocks (MK)	10	10	1
Sweep3D elapsed execution time [s]	121	505	129
Sweep3D measurement dilation [%]	1	5	3
Measurement unification time [mins]	2.5	43	0.2
Trace event records [G]	39	59	510
Trace buffer content (max) [MB]	2.75	2.75	27
Total trace size (uncompressed) [TB]	0.51	0.79	7.6
Trace (physical) files [#]	196,608	294,912	576
Trace open/create time [mins]	5	86	10
Trace flush rate [GB/s]	12.7	3.7	19.1
Trace analysis time (total) [s]	236	368	546
Trace timestamp correction [s]	25	—	—
Trace analysis replay [s]	93	11	74
Trace analysis collation [s]	30	162	91

Note that these measurements were taken during production on non-dedicated systems, where run-to-run variations (particularly with respect to I/O) can be considerable. The largests runs are one-off measurements taken at different times, where system (OS and I/O) configurations are continually upgraded and adjusted.

In the table, the Cray XT and initial BG/P measurement use the same Sweep3D application configuration (albeit with somewhat different numbers of processes) and comparable versions of Scalasca. Both Sweep3D and Scalasca are found to perform considerably better on the Cray XT5. The rightmost column used the improved Sweep3D configuration for BG/P and an enhanced Scalasca prototype for measurement. While the modified communication behaviour notably improves Sweep3D execution performance (bringing it largely into line with that on Cray XT5), with ten times the number of messages being traced, time for trace I/O increases commensurately. Scalasca measurement operations, however, for the unification of definition identifiers, opening (creating) the trace files, and writing the analysis report are not related to the number of messages and would take the same time in both configurations. Since these were prohibitively expensive with the original version of Scalasca, each of these aspects has been addressed in a new improved version of Scalasca.

Measurement unification refers to the processing of definitions (and corresponding identifiers) which were produced locally by each process, to create a unified set of global definitions and mappings for each rank. These allow the event traces from each process generated using local definitions to be correctly interpreted for an integrated analysis and avoids needing to re-write the trace files. A comparable unification is similarly required when collating an integrated analysis from the callpath profiles generated by each process with runtime summarization. Offline approaches to definition unification, processing files of definition records written by each process, were identified as the critical performance and scalability bottleneck for early Scalasca measurements, and replaced with online unification using MPI communication at the end of measurement collection. Although the performance was greatly improved, unification was still essentially sequential, unifying the definitions for each process rank in turn, and as a result the time was still substantial for very large-scale (and complex) measurements. With the introduction of a hierarchical unification scheme [16] in the latest development version of Scalasca, unification time for Sweep3D trace experiments with 294,912 processes on BG/P has been improved 200-fold from over 40 minutes down to under 13 seconds (and 60-fold for 196,608 processes on XT5 from 2.5 minutes down to less than 2.5 seconds).

The global high-resolution clock available on Blue Gene systems allows direct comparison of event timestamps on different processors, however, such a clock is not available on XT systems. In that case, Scalasca measurement uses sequences of message exchanges to determine clock offsets during MPI initialization and finalization and subsequently applies interpolation to adjust recorded timestamps prior to analysis. Since clock drifts are not constant, remaining logical clock violations

identified during trace analysis necessitate additional corrections. The time required to correct timestamps varies considerably according to the types and frequencies of violations encountered, such that for the series of measurements shown in Figure 2:a(ii) the trace from 24,576 processes which had many violations required much more processing than that from 49,152 processes which was fortunate to have none at all.

Although trace files are read and analyzed 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. Using a Scalasca prototype implementation, analysis report writing time was improved 8-fold for the BG/P `MK=1` experiment by dumping gathered metric severity values in binary format, while retaining XML for the metadata header [16].

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. (The Lustre filesystem on the XT5 apparently had no problem creating 196,608 files in only 5 minutes.) Employing the SIONlib library, 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, reduced the time for creation of the experiment archive directory and trace files down to 10 minutes.

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 (as will be seen in following examples). The graphs in the left column of Figure 2 show 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. Due to the ten-fold increase in the number of messages in the `MK=1` configuration, rather longer is required for processing MPI message sends and receives, however, waiting times are much less. In each case, most waiting time is found to be "Late Sender" situations, where a blocking receive is initiated earlier than the associated send as illustrated in Figure 5, 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.

4.3. *Scalasca Analysis Report Examination*

Figures 3 and 4 present Sweep3D execution analysis reports from the tracing experiments performed with Scalasca on *Jaguar* Cray XT5 and *Jugene* IBM BG/P with 196,608 and 294,912 MPI processes respectively.

In Figure 3, exclusive "Execution" time is selected from the tree-structured

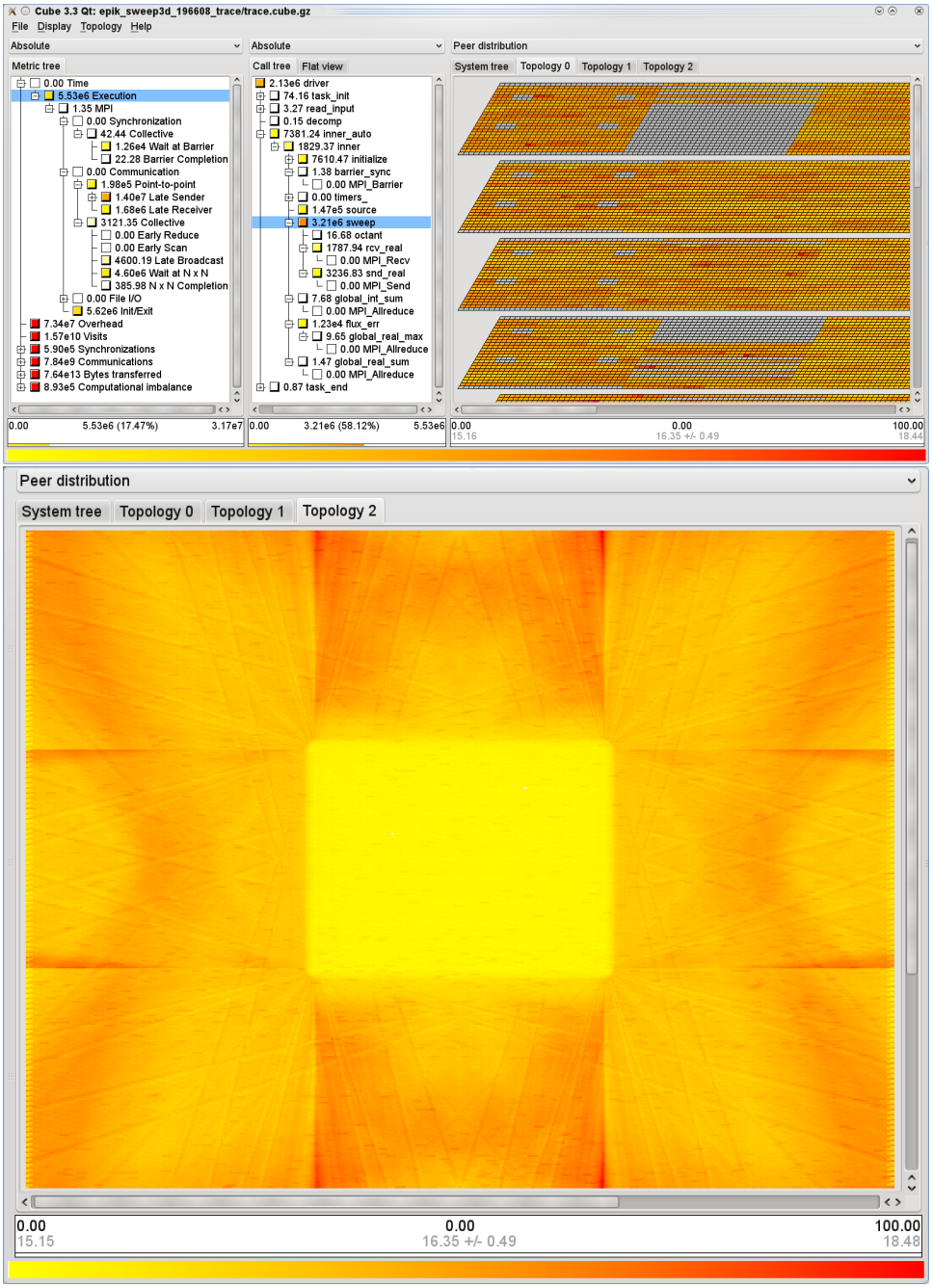


Fig. 3. Scalasca analysis report explorer presentations of a Sweep3D trace experiment with 196,608 processes on the *Jaguar* Cray XT5 system. Exclusive "Execution" time corresponding to local computation in the `sweep` routine and its distribution by process shown with part of the machine physical topology (where unallocated nodes in XT cabinets are shown grey), and below it an additional view of the imbalance using the application's 512×384 virtual topology.

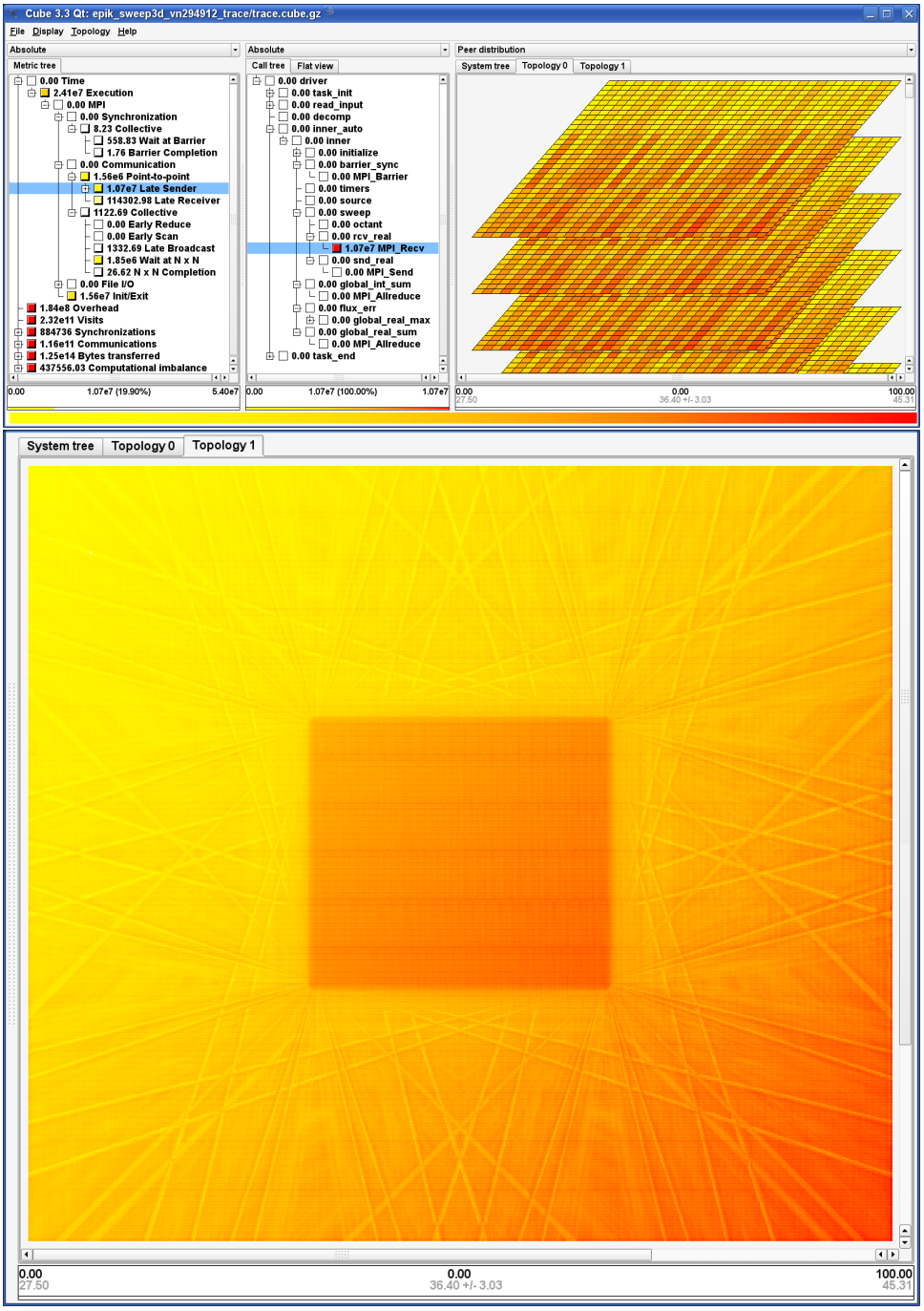


Fig. 4. Scalasca analysis report explorer presentation of Sweep3D execution performance with 294,912 MPI processes on *Jugene* IBM BG/P. The distribution of "Late Sender" waiting time metric values for the `MPI_Recv` callpath for each process are shown with the physical three-dimensional torus topology (top) and 576×512 application virtual topology (below).

performance metric hierarchy in the left panel, annotating the program call-tree in the middle panel with the local computation time for each callpath. With the callpath to the primary `sweep` routine selected, the distribution of metric values for each process are shown with the physical machine topology in the right panel. Additionally, the application's 512×384 virtual topology has been used for the lower display. Processes in the topology displays and the boxes next to nodes in the trees are colour-coded by metric value according to the scale at the bottom of the window.

17.5% of the total CPU time in the XT5 experiment is exclusive "Execution" time (with the remainder being broken down into various forms of MPI communication and synchronization time). 58% of this is computation in the key `sweep` routine itself (and not in paths called from it). Variation by process is found to be from 15.15 to 18.48 seconds (16.35 ± 0.49), and while there is no obvious correlation to the physical topology, a clear pattern emerges when the application's two-dimensional grid is used to present the computation time that is both regular and complex. The interior rectangular block has uniformly low computation time, with higher times in the surrounding border region and particularly on numerous oblique lines of processes radiating from the interior to the edges of the grid.

By comparison, Figure 4 is showing the "Late Sender" waiting time for point-to-point MPI communication on BG/P, which is 20% of the total CPU time in this case, where the improved `MK=1` configuration has been used. (In Figure 3, "Late Sender" time is 44%.) Values range from 27.50 to 46.31 seconds (36.40 ± 3.03) in a pronounced pattern similar to an inverse of that seen for the computation time. The central rectangular block has notably higher waiting time and the same intricate pattern of sharp oblique lines radiate from the central block to the edges, together

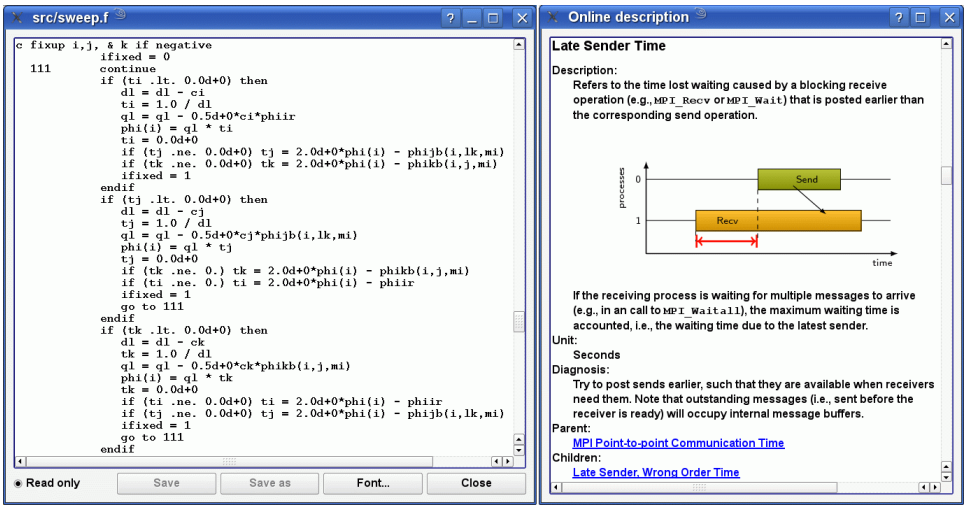


Fig. 5. Scalasca auxiliary windows showing the section of Sweep3D routine `sweep` source code for the flux correction 'fixups' (left) and explaining the "Late Sender" time metric (right).

with a background progressively increasing from the NW to SE corners. The computational imbalance has been superimposed with the progressive background pattern characteristic of the wavefront sweep communication. The computational imbalance has the same pattern and magnitude in both `MK=10` and `MK=1` configurations (though the total cost of the `sweep` routine is somewhat less in the latter), which subsequent analysis refined to the flux correction code shown in Figure 5. Whereas the communication waiting time dominates in the default `MK=10` configuration, due to insufficient parallelism in the wavefront sweeps, in the `MK=1` configuration waiting time is substantially reduced overall, yet notably amplified by the interaction with the computational imbalance to become a much larger proportion of the reduced computation time.

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 [13]. These Scalasca experiments on IBM BG/P and others on Cray XT indicate that the communication network and mapping of processes are not pertinent to the communication overhead and imbalance.

5. Refined Instrumentation and Analyses of SWEEP3D Execution

To isolate the origin of the imbalance, the Sweep3D source code was manually annotated with Scalasca instrumentation macros. Since the load imbalance was found not to differ in its characteristics on each platform or with larger scale, new experiments were done at the modest scale of 16,384 processes on BG/P.

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 5), as typically identified as a hotspot by sampling-based profilers (e.g., [14]). 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 compu-

tation time for applying these fixes is directly proportional to the number 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 6. 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 7. 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 6. (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 8. 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 6. 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 8 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 superimposes the underlying sweep with the additional computational imbalance.

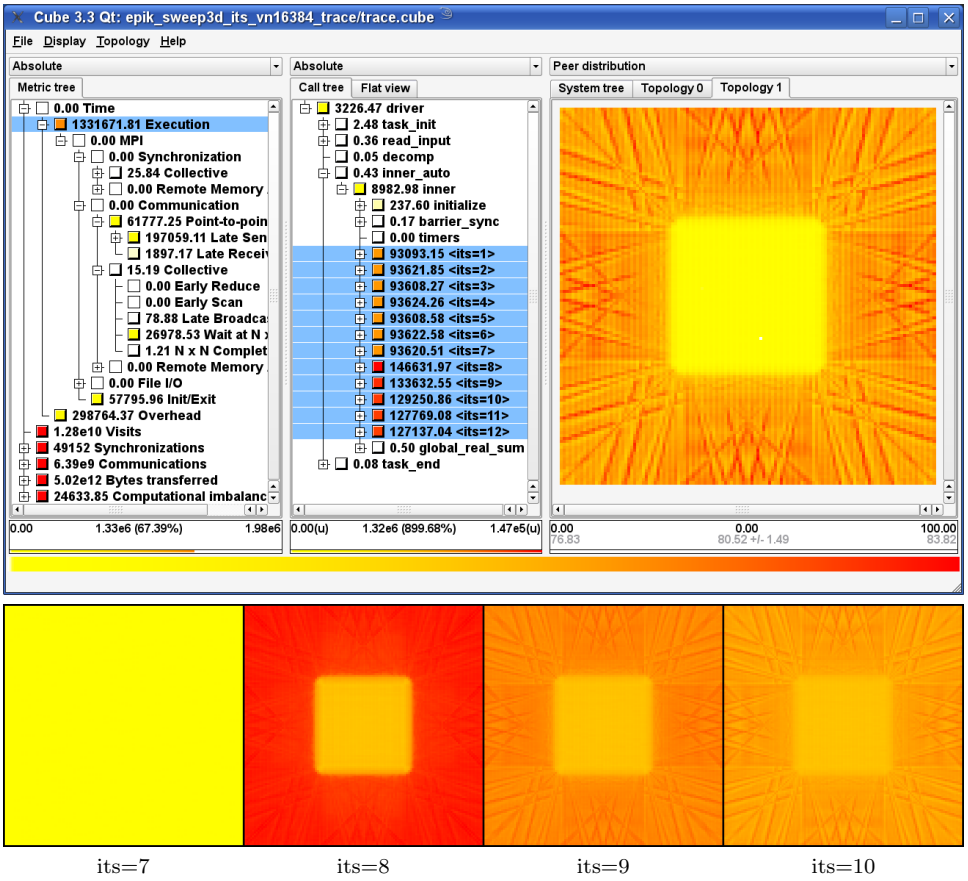


Fig. 6. Sweep3D computation *Execution* time variation by iteration (top) and 16,384-process distribution evolution for iterations 7 to 10 (bottom).

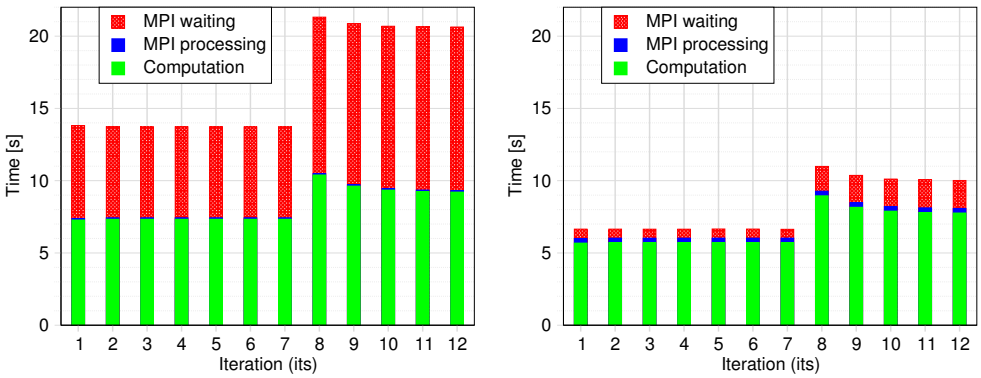
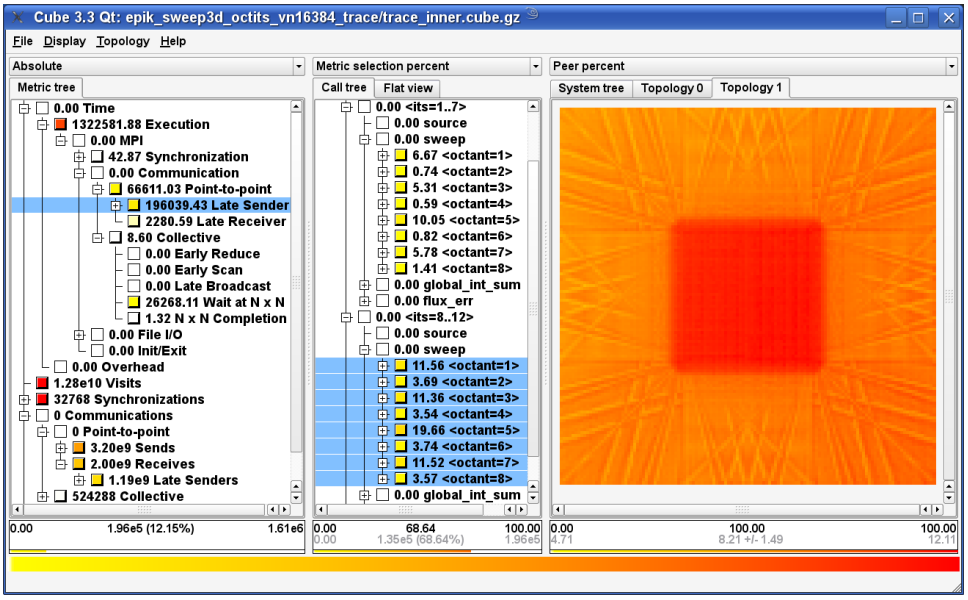
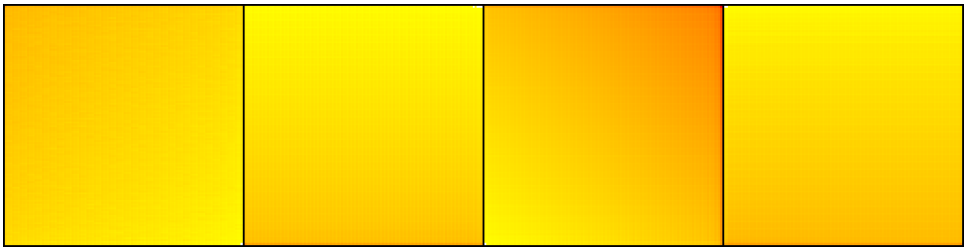


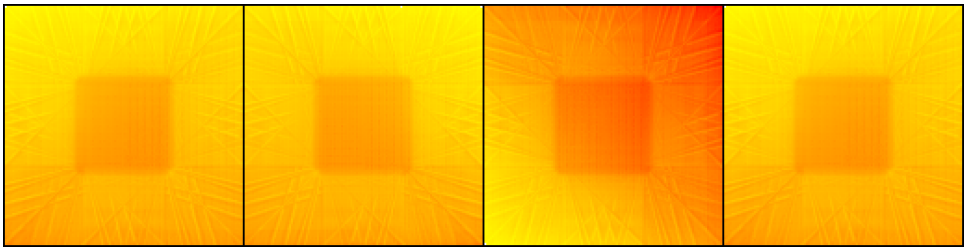
Fig. 7. Sweep3D iteration execution time breakdown with 16,384 processes on BG/P for `MK=10` (left) and `MK=1` (right) k-plane blocking factors.



Iterations without fixups:



Iterations with fixups:



octant=1+2 octant=3+4 octant=5+6 octant=7+8

Fig. 8. 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).

6. 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 modeled, 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. While flux corrections are necessary for a physically realistic solution, their computational expense and imbalance which disrupts communication in the wavefront sweeps, suggests that they should be used sparingly.

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 [17], and analyzing traces for the root causes of wait states to improve attribution of performance problems [18]. Tools for measuring and analyzing application execution performance also need to be highly scalable themself [19], as demonstrated by the Scalasca toolset with several hundred thousand Sweep3D processes on Cray XT5 and IBM BG/P, where multiple 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

This research was supported by allocations of advanced computing resources on the *Jugene* IBM Blue Gene/P of Jülich Supercomputing Centre at Forschungszentrum Jülich and the *Jaguar* Cray XT5 of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the US Department of Energy under contract DE-AC05-00OR22725.

References

[1] Jülich Supercomputing Centre, Jülich, Germany, "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*, vol. 22, no. 6, pp. 702–719, 2010.

[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] Cray Inc., "Cray XT," http://www.cray.com/Products/XT/Systems.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] W. Frings, F. Wolf, and V. Petkov, "Scalable massively parallel I/O to task-local files," in *Proc. 21st ACM/IEEE SC 2009 conference (Portland, OR, USA)*. IEEE Computer Society Press, Nov. 2009.

[16] M. Geimer, P. Saviankou, A. Strube, Z. Szebenyi, F. Wolf, and B. J. N. Wylie, "Further improving the scalability of the Scalasca toolset," in *Proc. PARA 2010 (Reykjavík, Iceland)*, (to appear).

[17] Z. Szebenyi, F. Wolf, and B. J. N. Wylie, "Space-efficient time-series call-path profiling of parallel applications," in *Proc. 21st ACM/IEEE SC 2009 conference (Portland, OR, USA)*. IEEE Computer Society Press, Nov. 2009.

[18] D. Böhme, M. Geimer, F. Wolf, and L. Arnold, "Identifying the root causes of wait states in large-scale parallel applications," in *Proc. 39th Int'l Conf. on Parallel Processing (ICPP, San Diego, CA, USA)*, Sep. 2010, (to appear).

[19] B. Mohr, B. J. N. Wylie, and F. Wolf, "Performance measurement and analysis tools for extremely scalable systems," *Concurrency and Computation: Practice and Experience*, vol. 22, Jun. 2010, (International Supercomputing Conference 2008 Award).