Portland State University

PDXScholar

Computer Science Faculty Publications and Presentations

Computer Science

2009

Evaluating Similarity-based Trace Reduction Techniques for Scalable Performance Analysis

Kathryn Marie Mohror Portland State University

Karen L. Karavanic Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/compsci_fac



Part of the Computer Sciences Commons

Let us know how access to this document benefits you.

Citation Details

Mohror, Kathryn Marie and Karavanic, Karen L., "Evaluating Similarity-based Trace Reduction Techniques for Scalable Performance Analysis" (2009). Computer Science Faculty Publications and Presentations. 220.

https://pdxscholar.library.pdx.edu/compsci_fac/220

This Technical Report is brought to you for free and open access. It has been accepted for inclusion in Computer Science Faculty Publications and Presentations by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

Evaluating Similarity-based Trace Reduction Techniques for Scalable Performance Analysis

Kathryn Mohror and Karen L. Karavanic Portland State University {kathryn,karavan}@cs.pdx.edu

ABSTRACT

Event traces are required to correctly diagnose a number of performance problems that arise on today's highly parallel systems. Unfortunately, the collection of event traces can produce a large volume of data that is difficult, or even impossible, to store and analyze. One approach for compressing a trace is to identify repeating trace patterns and retain only one representative of each pattern. However, determining the similarity of sections of traces, i.e., identifying patterns, is not straightforward. In this paper, we investigate pattern-based methods for reducing traces that will be used for performance analysis. We evaluate the different methods against several criteria, including size reduction, introduced error, and retention of performance trends, using both benchmarks with carefully chosen performance behaviors, and a real application.

1. INTRODUCTION

Today's high-end architectures contain tens to hundreds of thousands of processors, pushing application scalability challenges to new heights. Performance analysis is a necessary step to adapt codes to utilize a target high end machine. Correct diagnosis of certain complex performance problems that arise on high end systems requires detailed event traces. An "event" is a runtime occurrence of a program activity, such as a machine instruction or basic block execution, memory reference, function call, or a message send or receive. Generating event traces involves writing a time stamped record for each event, into a buffer or file for later analysis. Unfortunately, the collection of event traces presents scalability challenges: the act of measurement perturbs the target application; and the large volume of collected data increases the perturbation, and results in data files that are difficult, or even impossible, to store and analyze [24]. Several documented cases describe performance problems that appear only when the application is run at a large scale [18, 27], driving the need to be able to collect event traces for large runs. We have a conundrum: we need traces to correctly diagnose important performance problems, but the sheer volume of data collected makes collecting full traces at the very least prohibitive, and in the worst case impossible. For this reason, solving the scaling challenges of event tracing is an important problem for high end computing.

Given the challenges of tracing at the high end, one might be tempted to avoid it entirely. Profiling, for example, provides summary information and therefore exhibits better scaling behavior. However, the types of information provided by profiling are, in many cases, too limited for correct diagnosis of certain performance problems [7, 36]. An example of such a performance problem is "Late Sender" in a message-passing program. This is the situation where the receiving process waits at a blocking receive call waiting because the sending process hasn't yet reached the matching send call. While a profile could

indeed show that excessive time was being spent in receive operations, the data is not sufficient to distinguish between a late sender or some other root cause, such as network contention that caused the message to be received late. In contrast, an event trace captures the relative timing of events, and would show that the send operations started late and caused the receive operations to block. Tracing is also useful for showing the causality of events [31, 12]; the interactions between program elements, that can be difficult or impossible to understand from static analysis [22, 20]; and event patterns that reveal properties of programs, such as performance problems and locations of possible optimization [21].

One promising approach to highly scalable tracing is to filter or reduce the trace in some manner, either during or after the collection of trace records. Users who need to collect trace data currently resort to ad-hoc measures to reduce the amount of data collected; for example, tracing a reduced number of iterations of a loop. These measures have the potential to miss the performance problem altogether, e.g. if the problem doesn't occur during the measured iterations. One method for reducing the size of traces is to identify similar sections of a trace and retain only one representative of each pattern. However, determining the similarity between traces or sections of traces is not straightforward. The probability that any two trace sections will have exactly the same measurements is very small, so any similarity method will allow some amount of differences between similar traces. Despite this, it is critical that any differences allowed do not mask information needed for correct performance diagnosis.

Requirements for the accuracy and types of information in a trace vary based on the intended use: correctness testing and debugging, simulation, or performance analysis. Correctness testing and debugging generally only require that the trace retain the relative ordering of events that have the potential to affect each other: events within a single process or thread and synchronization events across processes or threads. For example, inspecting a trace of a parallel program could indicate the reason for a deadlock situation by showing the ordering of synchronization operations; a parallel program might hang because a process is waiting for a message that was never sent. Simulation requires traces that retain the order of events and possibly some timing information. Traces for simulation can be used to predict application performance on new or theoretical hardware. The events in the trace can be replayed using either averaged or predicted timing information for the new hardware. Generally, a single time value is used for all event occurrences instead of individual timing measurements for each event occurrence. For example, the average time to execute a send operation could be used as the time for all send operations in the trace. This tradeoff allows acceptable accuracy with faster time to simulated results and smaller trace files. Performance

analysis requires not only the relative ordering of events, but the timing information for individual events. Performance problems do not necessarily occur with a high degree of regularity, e.g. in every iteration of a loop, so individual event timings are needed to show the root causes of problems. For example, trace data can show a time-varying load imbalance in a parallel job, which causes some ranks to be late to a synchronization operation at varying times during the program execution. The individual event timings can show what events are taking more time in the slower ranks and in what iterations the slowness occurs.

In this work, our goal is to determine a similarity metric that yields adequate trace reduction and also retains the information needed for correct performance analysis. Achieving our goal required that we answer several key questions:

- What metrics can we use to evaluate and compare trace difference methods? In addition to file size reduction, we developed and used metrics for error, greatest possible file size reduction (i.e. potential for repeated patterns), and consistency of performance diagnosis.
- How much error should be allowed? Values that will likely never be exactly equal need to be compared. We had to decide how much each measurement can vary, and weigh the consequences of the amount of error. If we are matching traces for the purpose of trace compression, then a larger allowed error between traces would mean larger number of matches, and thus a smaller trace file. However, the larger error might prevent the correct performance diagnosis from being made.
- How can we measure the "goodness" of each approach? Most trace compression studies report the reduction of file size achieved; but no matter how much compression is achieved, if the reduced trace no longer contains the data needed for accurate performance diagnosis, the method is not useful for our purpose. We evaluate each approach not just on amount of compression, but also on amount of error and consistency of diagnosis, and discuss the tradeoffs in weighting the different metrics.

In this study, we perform a comparative evaluation of similarity metrics in current or proposed use for trace reduction. To evaluate the effectiveness of the similarity metrics, we apply the same trace reduction technique to full execution traces, varying the similarity method used to determine repeating patterns within the trace. Then we compare the results using three metrics: file size reduction, trace error, and retention of performance trends.

2. RELATED WORK

Previously proposed methods for reducing the sizes of traces for the purpose of performance analysis include deletion of similar trace sections; trace sampling; statistical clustering; and signal processing.

Knüpfer and Spooner define two sections of traces as similar if the call graph context and measurements of the events are equal. Knüpfer defines equality using both relative and absolute differences [19]; Spooner et al. use the relative difference in instruction counts [30]. Another approach defines similarity by event names. Chung et al. use a filter that detects repeated communication patterns [6]; they keep performance data for only one instance of each pattern. Freitag et al. use a periodicity detector to notice repeating sequences of events and keep a reduced number of iterations of each sequence [8]. Similarly,

Yan and Schmidt detect repeating sequences of events and store the average measurements of those events [36]. Noeth and Mueller also detect repeated sequences of message-passing events and store one copy of each sequence; they optionally store summary information about the events, such as average measurements [26]. In later work, they include the ability to store more detailed timing information: statistical "delta" times, histograms, or histograms by call sequence [28].

Other efforts use trace sampling to reduce trace size. Carrington et al. use trace sampling to reduce the amount of time it takes to gather memory reference traces for the purpose of performance modeling [3]. They collect data for a reduced number of executions of the basic blocks in a program. Vetter presents a method for statistically sampling MPI events [32]. Each time an MPI event is encountered, it is either sampled or not. For each sampled event, the tool can record statistics, log the event to a trace file, or ignore the data. Gamblin et al. use statistical sampling with a user-specified confidence interval and metric. [10].

Aguilera et al. [2], Nickolayev et al. [25], and Lee et al. [23] apply statistical clustering to traces and select a representative trace for each cluster of processes. Nickolayev and Lee use the Euclidean distance for clustering, while Aguilera uses a metric based on the amount of communication between two processes.

Several groups apply methods from signal processing to traces. Casas et al. and Huffmire et al. use the Haar wavelet transform to automatically determine the phases of a program [4, 16]. Gamblin et al. use the CDF 9/7 wavelet transform to compress traces collected for the purposes of detecting load imbalance [9]. Hauswirth et al. use dynamic time warping to decide when two traces are similar for aligning multiple traces [14].

Researchers have evaluated several methods for deciding the goodness of a particular trace similarity metric. To our knowledge, ours is the only comparative study of the methods to see what is most appropriate for the purposes of performance analysis. Ratn et al. use aggregate statistical measures, such as total time spent in a function, to evaluate their method [28]. Gamblin et al. compute a trace confidence measure to evaluate their trace sampling results, which is tells the percentage of time the mean trace of sampled processes is within an specified error bound of the mean trace of the full trace [10]. In their wavelet transform method, Gamblin et al. use a root mean square measure to estimate the error in reduced traces [9]. They also present qualitative results, showing a visualization based on a reduced trace compared with one from a complete trace. Yan et al. compare the measurements in their reduced trace against the real trace time stamp by time stamp and produce both a relative and absolute measure of the overall differences [35]. In addition, they also present whole program statistical measurements and visualizations for qualitative comparison.

3. TRACE REDUCTION

In this section we describe our approach for trace reduction. Section 3.1 details our trace segmentation technique, and Section 3.2 describes the different similarity metrics we use to compare segments. This paper focuses exclusively on intraprocess reduction, that is, reducing the size of each individual per-task trace. In practice these individual traces are first collected separately, then merged into a single trace file representing the entire application run. Therefore, reducing each

```
int main(){
       start segment("init");
       MPI Init();
       end segment("init");
       for(i=0; i < 100; ++i){
            start_segment("main.1");
            do_work();
            MPI Allgather();
            end segment("main.1");
       for (j=0; j < 10; ++j){
    start_segment("main.2");</pre>
             do_other_work();
             end segment("main.2");
             while(k < otherRanks){
                start_segment("main.2.1");
                MPI Sendrecv();
                end_segment("main.2.1");
       start_segment("final");
       MPI Finalize();
       end_segment("final");
```

Figure 1: Segment Context Marking. We show a single function, main() with the instructions added to mark the segment contexts. We mark initialization, finalization, and all loops. The segment context names are hierarchical: the second loop is marked "main.2" and its subloop is marked "main.2.1". Segment marking is automated using a dynamic instrumentation library.

per-task trace prior to merging will reduce the application trace accordingly.

3.1 Trace Collection and Segments

We collected full traces of time stamped function entries and exits for the benchmarks and application as follows. First we insert segment markers into the source code that are repeated in the trace during execution. We define segments as follows: the initial segment starts at entry to main; for each program loop containing at least one measured event, we stop the current segment before the loop starts, start a new segment at the top of each loop iteration, stop the segment at the bottom of the loop iteration, and start a new segment after the last iteration of the loop completes; and end the final segment at program termination. The segment context is the section of code, for example, the main.1 loop in Figure 1. We used the dynamic instrumentation library Dyninst [15] to instrument the full application for both function entry and exit tracing as well as inserting segment begin and end markers. The simple benchmarks were marked manually.

We compare the segments for each context pair wise to determine if they are similar. If they are, we say that the segments match and retain a single representative segment. Each segment s_i contains an ordered list of events $E_i = \{e_0, e_1, ..., e_m\}$. We maintain a list storedSegments, which contains the segments that represent the performance behaviors in the execution, and a list segmentExecs that holds the starting times and identifier of each representative segment so that we can later recreate a full trace. Given an equivalence operator \approx for some similarity metric, and a segment s_{new} that has events E_{new} the algorithm comparing segments is as follows:

```
For i = 0 to len(E_{new}):
    E_{new}[i].start = E_{new}[i].start - s_{new}.start
    E_{new}[i].end = E_{new}[i].end - s_{new}.start
s_{new}.end = s_{new}.end - s_{new}.start
match = False
For i = 0 to len(storedSegments):
    s_{stored} = storedSegments[i]
    match = compareSegments(s_{new}, s_{stored})
    If match = True:
       segmentExecs = segmentExecs \cup (s_{stored}.id.s_{new}.start)
       break
If not match:
    s_{new}.id = getNewId()
    segmentExecs = segmentExecs \cup (s_{new}.id, s_{new}.start)
    s_{new}.start = 0 \\
    storedSegments = storedSegments \cup s_{new}.
Boolean compare Segments (s_{new}, s_{stored}):
    If s_{new}.context \neq s_{stored}.context: return False
    If len(E_{new}) \neq len(E_{stored}): return False
    For i = 0 to len(E_{new}):
           If E_{new}[i].id \neq E_{stored}[i].id: return False
    If s_{new} \approx s_{stored}: return True
    Else: return False
```

Note that a segments match requires that segments have the same context and the same number of events occurring in the same order. We give examples of segment matching in Figure 2.

3.2 Similarity Metrics

We used several methods to decide the similarity of segments. Each of these is described below. Our choices were inspired by methods used by other researchers to reduce traces (See Section 2.). They fell into two categories: distance methods and iteration-based methods.

3.2.1 Distance Methods

The distance methods produce a difference measure, which is then compared against a user-supplied threshold to determine the presence or absence of a match. Several of the difference methods are standard methods for computing distances between values and sets of values. We use the relative difference (relDiff), absolute difference (absDiff), and three variations on the Minkowski distance (Manhattan, Euclidean, Chebyshev), and wavelet transforms (avgWave, haarWave).

relDiff. We compare the relative differences between each event measurement against a user-defined threshold; if greater, the events are not equal:

$$relDiff(x_1, x_2) = \frac{|x_1 - x_2|}{\max(x_1, x_2)}.$$

To see how *relDiff* matches segments, we consider our example in Figure 2. We compute the relative differences between each of the paired measurements in the segments. If any are above our chosen threshold, say 0.5, then the match fails. Comparing s2 with s1, we first compare the start times of the do_work event: x_1 =1 and x_2 =1, with relative difference 0. Since the relative difference is less than 0.5, we continue on computing relative differences. Next we check the end times for the do_work event. Here we compute a relative difference: x_1 =17 and x_2 =40, giving a relative difference of 0.58. This is above our threshold, so the segments do not match. When we compare s2

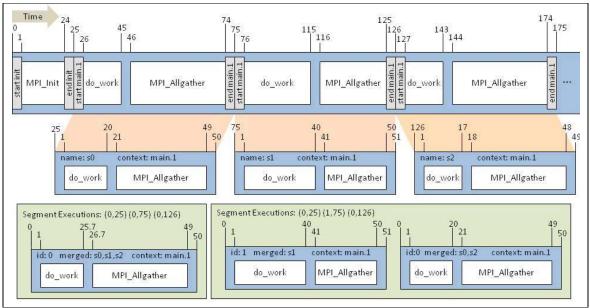


Figure 2: Trace and Segments Example. Here we show a portion of an example trace and three segments to illustrate segment matching. The top bar represents a portion of a trace for the program in Figure 1. Time increases from left to right, and time values are indicated above the bar. Segments markers are shown as light gray rectangles with vertical text that indicates the context of the segment. Events are shown in white boxes. Below the trace, we show the result of segmentation. In each of the three segments, the time stamps for the events and ending time of segments are adjusted relative to the start time of the segment. We name the segments s0, s1, and s2. In the bottom row, we show two examples of segment matching (See Section 3.2.).

with s0, we find that no differences are greater than 0.15 (x_1 =17, x_2 =20), so the segments match. The new segment is discarded since its behavior is reflected in the measurements in s0.

The relative difference function compares each measurement with its paired counterpart in isolation. The computed difference is proportional to the magnitude of the paired measurements, meaning that larger differences between larger measurements don't overshadow differences in smaller measurements. Because the difference between each measurement pair will be judged in isolation, the relative difference should be one of the strictest difference criteria in our set. The choice of threshold used will have a large bearing on the degree of matching, and hence on the reduction in file size.

One problem with relDiff appears when comparing time stamps in a series. For example, assume the threshold for comparing time stamps is 0.25. When we compare events that start at times 1 and 2, the relative difference is $\frac{2-1}{2} = 0.5$. This would result in a failure to match the events even though there is a difference of only one time unit between the events. In contrast, if we compare events that start at 100 and 125, the relative difference is 0.2, which is a match even though there is a difference of 25 time units. We expect relDiff to produce reduced traces with a low amount of error, but with less file size reduction.

absDiff. As with the relDiff, each measurement is compared with its counterpart. A fixed size difference, determined by a threshold, is allowed for each measurement pair. Using our example segments in Figure 2, and a threshold of 20, we see that s2 will not match s1, because the end times of do_work are 23 time units apart. However, there are no differences larger than 3

between s2 and s0, so those two segments match. The threshold choice has an impact on file size and accuracy. We expect this method to produce fairly accurate results, especially with respect to the timing of events across processes, because unlike *relDiff* it will not have an unfair bias towards events that occur later in the trace.

Manhattan, Euclidean, and Chebyshev. We compute the Minkowski distance between segments using the formula in Eq. 1. If the distance is greater than a user-specified threshold multiplied by the maximum value in the event measurements, then the events are not equal. The Manhattan, Euclidean, and Chebyshev distances are special cases of the Minkowski distance, with m equal to 1, 2, and $\lim_{m\to\infty}$ respectively [13]. The Chebyshev distance is defined to be the largest difference between two measurements.

Eq. 1

$$L_m = \left\{ \sum_{i=1}^{n} |x_i - y_i|^m \right\}^{1/m}$$

Using our example in Figure 2, to compare s2 and s1, we create a vector of the measurements for s2, (49, 1, 17, 18, 48), and one for s1, (51, 1, 40, 41, 50). The Manhattan, Euclidean, and Chebyshev distances between these vectors are 50, 32.6, and 23, respectively. The largest measurement in the pair of vectors is 51. If we choose a threshold of 0.2, then the highest the computed distance can be for a match is 10.2, so s2 and s1 will not match using any of the Minkowski distances. When we compare s0, (50, 1, 20, 21, 49), with s2, we get distances of 8,

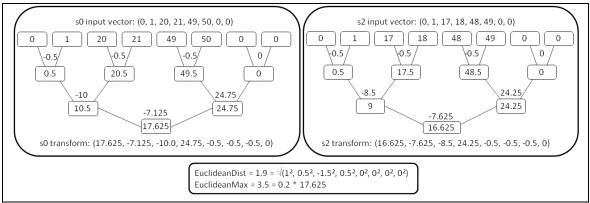


Figure 3: Wavelet Transform Example. Here we show two example average wavelet transforms. We iteratively compute averages (shown in boxes) and differences (shown between edges) for pairs of numbers, starting with the original vector. To compare the two transforms of s0 and s2, we compute the Euclidean distance between them and compare it against a threshold (0.2) multiplied by the largest element in the vectors (17.625).

4.5, and 3. The maximum value in the two vectors is 50, so the highest the distances can be for a match is 10. This means that s2 would match s0 for each of these distance metrics.

There are several issues to consider for the Minkowski distances:

- As m increases in the Minkowski distance (See Eq. 1.), the influence of the larger differences increases, and the influence of the smaller differences decreases. In the extreme case of the Chebyshev distance, only the maximum difference has any bearing on the distance value. As the number of measurements being compared increases, the values of the Manhattan and Euclidean distances increase. Given vectors of constant differences greater than 1, the Manhattan distance increases quite rapidly linearly, and the Euclidean distance increases in the manner of √x. If the differences are all between 0 and 1, the computed distances increase more slowly.
- When time stamp values are being compared, e.g. start time
 and end time for events, the values are always increasing
 within a segment. This means that longer segments are
 judged less critically than shorter segments, because the
 maximum values that are compared with the distance
 measurement are larger.

Based on these trends, we expect that the Manhattan distance would give the most accurate results, because it gives larger weight to the smaller differences. The Euclidean distance would give slightly less accurate results, given the bias towards larger differences. The Chebyshev distance would be least accurate, because it only accounts for the largest difference measure.

Wavelet transform. The discrete wavelet transform iteratively decomposes a signal of size L into two subsignals of size L/2. The first L/2 values give the trends in the original signal, and the second L/2 values give the fluctuations. Intuitively, it computes the averages and differences between pairs of numbers [17]. We give examples of transformations in Figure 3.

We use two wavelet transforms in our experiments: the average transform described in Figure 3 (avgWave), and the Haar transform (haarWave). The Haar transform is very similar to the average transform, with the only difference being that the averages and differences are multiplied by $\sqrt{2}$ [33]. For

example, the trends computed in step 3 in Figure 3 would be $(9\sqrt{2}, 24.25\sqrt{2})$. For our implementation, we construct a vector for each of the segments to be compared. The first element of each vector is the relative start time of the segment, which is 0 in all cases. This is followed by the event entry and exit time stamps for all events in the segment. The last element is the exit time of the segment. Both transforms require an input vector with a length that is a power of two. We allocate space for the vector so that its length is the next power of two after the number of time stamps in the vector. We zero-pad the vector after the last time stamp element to the end. To compare transformed vectors, we compute the Euclidean distance between them [5] and compare it against a threshold multiplied by the largest value in the pair of transformed vectors. In Figure 3, we show an example comparison of the segments s0 and s2 from Figure 2. Because the computed Euclidean distance, 1.9, is less than the maximum allowed, 3.5, s0 and s2 match.

For both transforms, the values in the transformed vectors will be smaller than the values in the original vectors. The Haar transform has several properties that the average transform does not, including preservation of the Euclidean distance [5]. However, its values will be larger than those of the average transform since all values are multiplied by $\sqrt{2}$. For the Haar transform, we expect more accurate results than from the Euclidean distance because the maximum value in the transformed vector will be smaller than the maximum value in the original vector, so the threshold test will be stricter. The values in the vector from the average transform will be smaller still; however, the Euclidean distance is not preserved, so the potential exists for a less strict test than the Euclidean distance.

3.2.2 Iteration-based Methods

We chose two iteration-based methods: *iter_k* and *iter_avg*.

iter_k. Only keep a fixed number of each traced segment of code. We expect this method to produce small data files. For our example in Figure 2, if we chose k=3, we would keep all three copies of the main.1 segment in the list of stored segments. However, if k=2, then we would keep s0 and s1 and discard s2.

iter_avg. Keep the average measurements for each traced section of code. We expect this method to produce the smallest data sizes, since segments with the same context and same

events will always match. To illustrate this method, we use the segments in Figure 2 and the stored segments scenario on the left. For this method, we never have more than one copy of the main.1 segment, and end up with a single copy of the main.1 segment that contains averages of the values of s0, s1, and s2.

We expect that these methods will produce fairly accurate data for applications that have little behavior variability, but poorly for applications that do have performance variabilities.

4. EVALUATION METHODOLOGY

In this section we detail our framework for the evaluation of similarity metrics. We investigate traces collected for a set of benchmarks with known behaviors, and for a full application, running on a Linux cluster. Our evaluation focuses on three metrics: file size reduction, amount of error in the trace, and retention of performance trends. For file size reduction we simply compare the sizes of the reduced traces to the full-sized traces from which they were derived. We calculate the trace error by recreating an approximated full-sized trace from the reduced version, then comparing it to the actual full trace. We evaluate retention of performance trends by feeding the actual and approximated full traces into a performance analysis tool and examining any differences in the results.

4.1 Benchmarks

We crafted our benchmarks to represent classes of performance behaviors that occur in parallel programs on high end systems. These performance behaviors can appear with a high degree of regularity, sporadically, or progressively change over the iterations in the execution. To reflect this, we created a set of regularly behaving benchmarks, a set of irregularly behaving benchmarks, and a benchmark that simulates dynamic load balancing. Because we know the behavior patterns in each benchmark, we can evaluate how well each of the methods retains the performance behaviors.

We used the APART Test Suite (ATS) to create our benchmarks. The ATS a collection of utilities designed to create programs with known behavior for testing parallel performance tools [11]. We chose behavior patterns from the ATS that represent performance problems that require trace data for correct diagnosis. For parallel programs, these performance behaviors fall into four categories based on the communication pattern being used. We describe these communication patterns here using MPI functions as examples.

- N →1. N processes send data to 1 process. If any of the sending processes are late, then the receiving process blocks, waiting for them to execute the send operation. Example MPI functions for this pattern are MPI_Reduce and MPI_Gather, with corresponding performance behavior problems early_reduce and early_gather.
- 1 → N. I process sends data to N processes. If the sending process is late, then all N receiving processes will block until the send is executed. Example functions are MPI_Bcast and MPI_Scatter. The corresponding performance problems are late_broadcast and late_scatter.
- 1 →1. I process sends to 1 process. There are two cases. In the case of a non-blocking send and a blocking receive, if the sending process is late, the receiving process will block. In the case of a synchronous send, the sending process will block if the receiving process is late. Example communication routines are MPI_Ssend and MPI_Recv,

- with corresponding performance problems *late_receiver* and *late_sender*.
- N →N. N processes send to N processes. Here, all N processes depend on all other processes involved in the communication to proceed. If any of the N are late, then the rest of the processes block until all have reached the communication routine. An example is MPI_Barrier with corresponding performance problem imbalance_at_barrier.

Benchmarks with Regular Behavior. We chose five example benchmarks provided with ATS with regular behavior: early_gather, imbalance_at_mpi_barrier, late_receiver, late_sender, and late_broadcast. Each of the benchmarks simulates a program with the given behavior problem with the same severity in each iteration. In other words, all iterations of each program will exhibit the performance problem and all iterations should be very similar. All runs had 8 processes.

We expect the similarity methods to do relatively well on this set of benchmarks since the iterations have regular behavior. They should be able to find a large number of segments matches and still retain the correct performance behaviors.

Benchmarks with Irregular Behavior. For this category, we used ATS to create new benchmarks with irregular behavior. The benchmarks simulate the system interference identified by Petrini et al. when they ran an application on ASCI Q [27]. The system interference prevented the application from scaling as predicted. The benchmarks contain iterations with work periods that last approximately 1 ms followed by a communication step, using the communication patterns described previously. The load for each process is constant in each iteration and across processes: the only performance problem comes from the interference. We simulated the system noise using timers to interrupt the processes as described by Petrini et al. We used two simulation scenarios. The first was a 32-process run, with each of the 32 processes simulating the interrupts specific to the 32 nodes in an ASCI Q cluster. The second was also a 32-process run, but with the simulated amount of system interruptions that would occur if there were 1024 processes in the run. When we refer to the benchmarks in the first category, we use the communication pattern and either a _32 or a _1024, to indicate whether 32 or 1024 processes were simulated, respectively.

For these benchmarks, we expect the methods to find a high number of matches, since most iterations are very similar. However, it will be important that they don't falsely match undisturbed and disturbed iterations, as this has the potential to mask or amplify the periodic behavior changes due to the simulated interruptions.

Dynamic Load Balancing. Here, we used ATS to create a program that simulates an application that does dynamic load balancing. For this benchmark, the performance of the iterations starts at about 1 *ms* and gets progressively worse, with one-half of the processes doing more work each iteration and the other half doing less work in each iteration, until the "load balancer" is triggered. The "load balancer" readjusts the amount of work on each processor to be equal. The performance problem exhibited by this program is *imbalance at mpi all to all*, which falls in the N-to-N communication category. This benchmark is referred to as *dyn load balance* and was run with 8 processes.

For this benchmark, we expect less overall matching since behavior changes with each iteration and very close performance behaviors reoccur only after each simulated load balance. Here it will be important that the similarity methods do not match segments with larger differences because the load imbalance may no longer be apparent in the reduced trace.

4.2 Application

We chose Sweep3D 2.2b, a structured mesh application that computes a 1-group time-independent discrete ordinates three-dimensional Cartesian geometry neutron transport problem [1]. Structured mesh applications have a regular partitioning of the data, where all interior data blocks have equal numbers of neighbors. It is likely that the performance will be very regular over the course of the program, which means that the reduction methods should be able to find a large number of segment matches without introducing a large amount of error. We collected traces for two runs of this application: an 8-process run with input file input.50, sweep3d_8p; and a 32-process run with input input.150, sweep3d_32p.

4.3 Evaluation Criteria

We chose four criteria to evaluate the metrics: percentage of full trace file size, degree of matching, approximation distance, and retention of correct performance trends.

4.3.1 Percentage of Full Trace File Size

We present the savings in file size as a percentage of the full, non-reduced trace file, as a relative measure of size reduction.

4.3.2 Degree of Matching

The degree of matching metric is a measure of how many segment matches occurred. We define it to be the ratio of the number of matches to the number of possible matches. The number of possible matches is limited by the structure of the program. For example, some portions of the code may only execute one time, e.g. an initialization step, and will not match any other event sequence in the trace. A possible match between segments exists if: the segments represent the same code location; they contain the same events in the same order; and all message passing calls and parameters are the same.

4.3.3 Approximation Distance

We estimate the error in the trace by recreating a full trace from the reduced trace and comparing each time stamp with its counterpart in the original full trace. The approximation distance metric tells what absolute difference 90% of time stamps had compared to the originals.¹

4.3.4 Retains Correct Performance Trends

Arguably, the most important criterion for evaluating a trace matching metric for the purposes of performance analysis is deciding whether or not the reduced trace still indicates the same performance problems as the full trace. For example, if an analyst inspecting a full trace detects a late sender performance problem, the same problem should be detected in the reduced trace with approximately the same severity. The KOJAK tool set

When recreating full traces for the iter_k method, we used the last segment that executed of each pattern to fill in the segment executions that were not collected. Alternatives include using the average measurements from the k collected segments, or using the centroid of those k segments as determined by a clustering algorithm. was developed to aid parallel performance analysts in the challenging task of performance diagnosis [34]. KOJAK's EXPERT tool reads in a trace file and produces a data file containing performance diagnoses. Each diagnosis consists of a metric, a code location, and a severity for each thread in the run [29]. KOJAK's CUBE tool reads in the analysis data and presents a visualization to the user, indicating the most important performance trends in the trace in a hierarchical manner.

We use the CUBE visualization tool to compare the performance diagnoses for the recreated traces against the diagnoses for the full trace (See Figure 4.). We determine whether a performance analyst would come to the same conclusions about the reduced trace as the full trace. If not, then the reduced trace is not adequate for performance analysis. We admit that this is a subjective test; however, we followed a set of guidelines when deciding if the diagnoses were sufficiently similar, so all the methods were subjected to the same criteria.

5. EVALUATION STUDIES

In this section, we present the results of two studies evaluating the similarity methods using the criteria and programs described in Section 4. We first present a threshold study for the similarity methods from the distance metric category. From this study, we choose a threshold for each of these methods that represents the best tradeoff in terms of file size reduction, measurement error, and retention of performance trends. In the second study, we present the results of a comparative study of the similarity methods, using the thresholds found to be best for each method in the threshold study.

5.1 Threshold Study

We investigated the behavior of the methods in reducing the traces of the benchmarks while varying the thresholds that determine whether two given segments should match or not match. The thresholds for relDiff, Minkowski distances, and the wavelet transforms were 0.1, 0.2, 0.4, 0.6, 0.8, and 1.0. The thresholds for iter_k were 1, 10, 50, 100, 500, and 1000, and for absDiff were powers of 10 from 10¹ to 10⁶. Since no thresholds are used with the iter_avg method, it was not included in this study. The criteria we used to evaluate the methods were file size, approximation distance, and retention of performance trends (For file size reduction and approximation distance, see Figures 10-16 in the Appendix for the benchmarks and Figures 17-19 for sweep3d. For retention of performance trends, see Tables 1-18 in the Appendix.). For each method, we chose a representative threshold to be used when comparing the methods against each other.

relDiff. The file size for each benchmark and the sweep3d runs decreased relatively steadily with increasing threshold. The approximation distance remained small until the 0.8 threshold, after which there was a large jump for many of the benchmarks and sweep3d_32p. Performance trends were correctly retained for most programs up to a threshold of 0.8. Based on the jump in approximation distance and loss of performance trends after threshold 0.8, we chose 0.8 as the best threshold for relDiff.

absDiff. Here the file sizes for the benchmarks and sweep3d dropped off fairly quickly at a threshold of 100 and continued to decrease slightly with increasing threshold. The approximation distance stayed relatively low up to a threshold of 10⁴, after which there was a sharp increase for several of the benchmarks and sweep3d_32p. Performance trends were retained for most

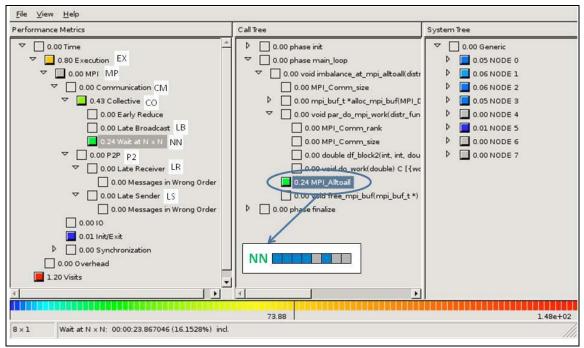


Figure 4: KOJAK Performance Analysis and Derivation of Our Performance Diagnosis Representation. Here we show a screenshot of KOJAK's EXPERT tool displaying the performance diagnosis for dyn_load_balance. The color bar on the bottom shows the severity levels, with blue being low and red high, and gray indicating 0 or close to 0. The left panel shows the performance metrics; the middle panel shows the code locations; and the right panel shows the processes. The color blocks next to each metric, code location, and process show the severity for the selected combination. Above, we have selected the function MPI_Alltoall and the "Wait at NxN" metric. This combination has green or "medium-low" severity and the severity is close to 0 for ranks 4, 6, and 7 and fairly low for ranks 0-3 and 5. We represent this diagnosis by abbreviating the metric name, e.g. NN for "Wait at N x N," coloring the metric abbreviation according to the severity indicated in the code location pane, and coloring squares for each process according to their severity levels. White squares indicate negative severities. We show the abbreviations we use for selected KOJAK metrics in white rectangles next to the metric names.

programs at a threshold of less than 10³. Because the file sizes were relatively low and performance trends were retained at 10³, we chose 10³ as the representative threshold for *absDiff*.

Manhattan, Euclidean, and Chebyshev. When observing file sizes changes, the Manhattan and Euclidean methods behaved quite similarly; the Chebyshev method showed some differences. For the Manhattan and Euclidean methods with the regular benchmarks, the 1-to-1 irregular benchmarks, and sweep3d, file sizes decreased relatively steadily with increasing threshold; with the other irregular benchmarks, the file size decreased only slightly with increasing threshold, because a matching that was close to optimal was reached early, at a threshold of 0.1. For Chebyshev with the 1-to-1 irregular benchmarks and sweep3d, file size decreased with increasing threshold; with the regular benchmarks and remaining irregular benchmarks, file size was relatively constant with increasing threshold. For all three methods, we observed the following behavior in approximation distance: with the regular benchmarks, approximation distance was relatively constant with increasing threshold; with the 1-to-1 irregular benchmarks, approximation distance increased with increasing threshold; with the remaining benchmarks, the approximation distance remained low until after the threshold of 0.8, after which there was a large jump. For sweep3d and Manhattan and Euclidean, approximation distance increased with increasing threshold; for Chebyshev, the approximation distance was small and relatively

constant until after the 0.8 threshold. For retention of performance trends, the Manhattan distance did well up to a threshold of 0.4, and the Euclidean and Chebyshev distances did well up to 0.2. We based our selection of best thresholds for these methods on the retention of performance trends metric, because we consider this metric to be the most important. We chose 0.4 as the best threshold for the Manhattan distance and 0.2 for the Euclidean and Chebyshev distances.

Wavelet Transforms. For all evaluation criteria, avgWave and haarWave performed similarly. For all programs, file sizes decreased with increasing threshold, up to the point of perfect matching, after which no further decrease in size is possible. The best threshold in this category appears to be 0.4 for both methods, because file size decrease levels off after this threshold. The approximation distance for both methods remained steady with increasing threshold for the regular benchmarks and the irregular N-to1, N-to-N, and 1-to-N benchmarks. The approximation distance increased with increasing thresholds for the irregular 1-to-1 benchmarks and sweep3d. The threshold 0.2 is best for approximation distance. because of the relatively higher values for the dyn_load_balance benchmark and sweep3d after this threshold. For the majority of programs, performance trends were retained for both methods at thresholds below 0.2. For these reasons, we chose 0.2 as the best threshold for the wavelet transform methods.

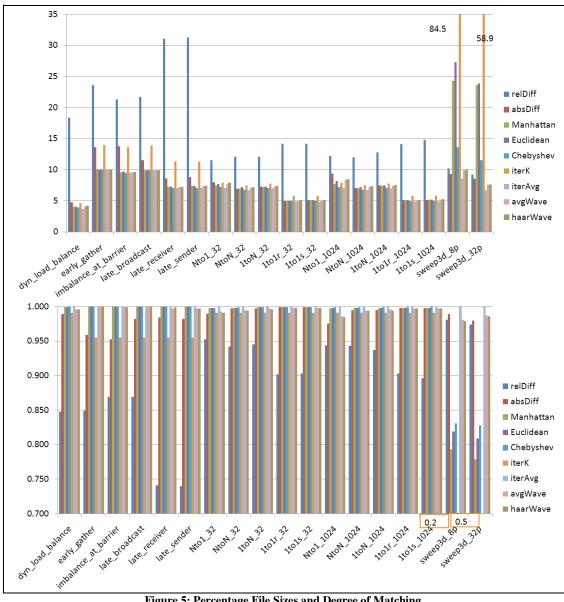


Figure 5: Percentage File Sizes and Degree of Matching.

iter_k. Generally speaking, there was an increase in file size and decrease in approximation distance with increasing k. Performance trends were retained for must programs up to threshold 10. The choice for the best k wasn't clear, but we chose k=10 as the best because the performance trends were retained for most programs at this threshold.

5.2 COMPARATIVE STUDY

In this section, we present comparative results for the different methods using size and degree of matching; approximation distance; and retention of performance trends as the evaluation criteria. Based on the results of the threshold study in Section

5.1, we present results for the best performing threshold for each method: 0.8 for relDiff, 1000 for absDiff, 0.4 for Manhattan, 0.2 for Euclidean and Chebyshev, 10 iterations for iter_k, and 0.2 for avgWave and haarWave.

5.2.1 Size and Degree of Matching

We present the data for reduction of traces for each method in Figure 5. The iter_avg method gives the best case values for this category, since exactly one segment is retained per loop with this method.

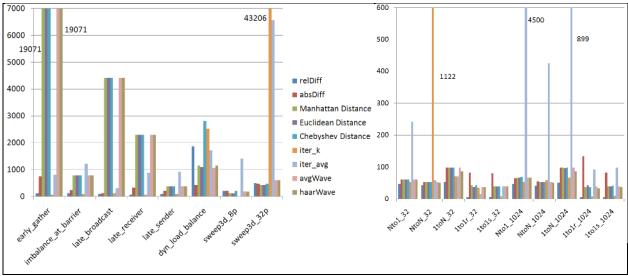


Figure 6: Approximation Distance Results for All Methods at Default Thresholds.

The benchmark data shows that for the most part, the degree of matching for each of the methods is greater than 0.9, meaning that greater than 90% of the segments were matched. Exceptions occur with relDiff, which had degree of matching scores as low as 0.74. RelDiff had the highest file sizes and lowest degree of matching scores. The next largest file sizes are generated with the iter_k method; however, they are not much higher than those for the other methods. The Minkowski distances, avgWave, and haarWave all have nearly identical results, with Chebyshev having a very slight advantage over the others. AbsDiff had only slightly larger file sizes than the Minkowski distances.

For sweep3d, the results are somewhat different. Because this application has very regular behavior, we expected the results to be similar to those of the benchmarks. However, because of the program structure, there are more segments, as well as differences within the segments, e.g. message passing parameters, that cause segments not to match. We see that *iter_k* performed the worst, with the highest file sizes and lowest degree of matching scores. This is because *iter_k* needed to keep 10 copies of each individual segment, regardless of how similar in performance they actually were, whereas the high degree of matching often results in fewer than 10 copies. The next worst performing were the Minkowski distances, again with *Chebyshev* having the smallest file sizes. The wavelet methods performed best, followed by *absDiff* and *relDiff*, each with very close to perfect matching and lowest possible file sizes.

The obvious best method in this category is *iter_avg*, since all segments match by definition. A comparison of the average file sizes for each of the other methods yields the following ranking: avgWave, haarWave, Chebyshev, absDiff, Manhattan, Euclidean, iter_k, relDiff.

5.2.2 Approximation Distance

Figure 6 shows the approximation distance results for each of the methods. High values for iter k and iter avg mean that there is irregularity in the execution that is not being captured in the iterations that are retained. High values for absDiff give a rough indication of the absolute difference of time stamps from the true values in the full trace. High values for the Minkowski and wavelet methods mean that there are high maximum values in the set of values being compared, relative to the distance between those values.

The methods show similar trends across the benchmarks with regular behavior. The <code>relDiff</code>, <code>absDiff</code>, <code>iter_k</code>, and <code>iter_avg</code> methods have consistently low values. The Minkowski distances, <code>avgWave</code>, and <code>haarWave</code> transform behave similarly, and have the highest values overall. The results for the <code>dyn_load_balance</code> benchmark show a different set of behavior, with <code>absDiff</code> having the lowest value, followed by <code>avgWave</code>, <code>Euclidean</code>, <code>Manhattan</code>, and <code>haarWave</code>. The interference benchmarks had lower overall approximation distance values than the other benchmarks, with similar results across the benchmarks. The worst performing methods in this case were <code>iter_avg</code> and <code>iter_k</code>. However, the approximation distance values are low in comparison to those for the other set of benchmarks.

The results for sweep3d show *iter_avg* performing the worst for the 8-process run, and *iter_k* and *iter_avg* the worst for the 32-process run, indicating that there are performance behaviors not being captured by those two methods.

The methods that performed the best in this category are *relDiff*, followed by *absDiff*, and then *iter_avg*. The rest of the methods allowed significant error into at least one of the reduced traces.

5.2.3 Retention of Performance Trends

We present summaries of the performance diagnoses given by KOJAK for selected benchmarks in Figures 7 and 8. We show how we derive the performance diagnoses charts and abbreviations for metric names in Figure 4. For the benchmarks with regular behavior, nearly all the methods performed quite

		MP	I_Alltoal	1		do_work
no loss	EX	MP	CM	СО	NN	EX
relDiff	EX	MP	CM	СО	NN	EX
absDiff	EX	MP	CM	СО	NN	EX
Manhattan	EX	MP	CM	СО	NN	EX
Euclidean	EX	MP	CM	СО	NN	EX
Chebyshev	EX	MP	CM	СО	NN	EX
iter_k	EX	MP	CM	СО	NN	EX
iter_avg	EX	MP	CM	СО	NN	EX
avgWave	EX	MP	CM	СО	NN	EX
haarWave	EX	MP	CM	CO	NN	EX

Figure 7: KOJAK Performance Trends for dyn_load_balance For Each Method at Default Thresholds. Here we show the results for each reduction method in the MPI_Alltoall and do_work functions. The first row shows the diagnoses for the full trace. Each box in a row shows a performance diagnosis for a single combination of metric and code location.

well. For late_receiver, all methods except <code>iter_avg</code> performed equally well, with all performance trends retained. The results for <code>iter_avg</code> with late_receiver showed differences significant enough that they may lead to an inaccurate performance assessment. For early_gather, all but the Minkowski distances, <code>avgWave</code>, and <code>haarWave</code> retained the correct performance trends. The results for imbalance_at_barrier showed that the Minkowski distances, <code>absDiff</code>, <code>iter_avg</code>, <code>avgWave</code>, and <code>haarWave</code> retained the performance trends, while <code>relDiff</code> and <code>iter_k</code> both showed a negative value for the major performance diagnosis. The amount of error introduced into the reduced traces caused time stamps to be skewed enough that the performance diagnoses resulted in negative values.

We show the major performance trends for dyn_load_balance in MPI Alltoall and do work as reported by the KOJAK tools for the full trace and all methods in Figure 7. The results for the no loss trace clearly indicate that the lower ranks are spending more time in MPI Alltoall, because the upper ranks are spending more time in do work. None of the methods gave perfect results for the dyn_load_balance benchmark; however, absDiff, Manhattan, Euclidean, avgWave, and haarWave gave the closest performance diagnoses because for the most part they maintained the performance differences due to load imbalance between the upper and lower ranks. Although Manhattan, Euclidean, avgWave, and haarWave lost the disparity in do work, the diagnosis "Wait at NxN" is nonnegative and maintains the disparity in behavior. AbsDiff maintained the disparity in performance in do work, but reported that "Wait at NxN" was negative. All other methods lose the expected disparity in do work.

For the interference benchmarks, all methods did pretty well on the N-to-1 and 1-to-N benchmarks, with the exception of *iter_avg*, which failed on three benchmarks, and *Chebyshev*, which failed on Nto1_1024. *AbsDiff* did less well on the 1-to-1 and N-to-N benchmarks. We show the data for 1to1r_1024 in Figure 8. *AbsDiff* picked up on the variations in the iterations due interference, which caused some performance diagnoses to be skewed in a positive or negative direction. The best performers for these benchmarks were *Manhattan*, *Euclidean*, and *avgWave*, followed by *relDiff*, and *haarWave*. *AbsDiff* and

iter_avg both only showed correct diagnoses for one benchmark, 1to1r_32 and 1to1s_32, respectively.

For sweep3d_8p and sweep3d_32p, all methods but *iter_avg* and *iter_k* produced correct data. *Iter_k* showed a non-existent disparity in rank performance in pmpi_recv in sweep3d_8p and a greatly inflated severity in pmpi_recv in sweep3d_32p. *Iter_avg* showed a much lower severity in sweep_ than did the no-loss trace for both sweep3d_8p and sweep3d_32p.

The best methods in this category were *Manhattan*, *Euclidean*, and *avgWave* which correctly diagnosed 17 out of the 18 execution traces. HarrWave did second best, correctly diagnosing 16. The rest of the methods in order were: *relDiff* (14); *absDiff* and Chebyshev (13); *iter_k* (12); and *iter_avg* (6). The relatively poor performance of *iter_k* in this category could be due to our choices in implementing this method¹. It is possible that the first iterations are more subject to variabilities in execution, before the processes synchronize into their regular behavior patterns, and that the last segment is not the best choice as a fill in for missing segments. *AbsDiff* seemed to amplify differences in the traces with interference, while *iter_avg* seemed to smooth out behavior patterns.

5.2.4 Discussion

For relDiff, we expected low error and relatively large files, which is exactly what we found to be true. For absDiff, we expected low error. We did find that absDiff had lower error when compared to most methods. We expected the Minkowski distances would favor long segments and error would be lowest for Manhattan, followed by Euclidean, and highest for Chebyshev. While we did definitely see more error in the traces produced by the Chebyshev method, the differences in the results for the Manhattan and Euclidean methods were largely undistinguishable. We expected iter k and iter avg to produce low error traces for programs with regular behavior and for iter_avg to have the lowest overall file sizes. We indeed found that iter_k did well for regularly behaving programs and less well for programs with varying behavior patterns. Iter_avg produced better results for the regular benchmarks than the irregular ones; the averaging of measurements tended to cause loss of information needed for diagnosis. For avgWave and haarWave, we expected stricter comparisons than Euclidean.

		MPI_Ssend							M	PI_Rec	v			do	work	
no loss	EX		MP	CM*	P2		LR	EX		MP	СМ	P2		LS	EX	
relDiff	EX		MP .	СМ	P2		LR	EX		MP	СМ	P2		LS	EX	
absDiff	EX			СМ			LR	EX	101	MP III	СМ	P2	101	LS	EX	
Manhattan			MP III	СМ	P2		LR	EX		MP	СМ	P2		LS	EX	
Euclidean	EX		MP	СМ	P2		LR	EX		MP	СМ	P2		LS	EX	
Chebyshev	EX		MP	CM T	P2		LR	EX		MP	CM T	P2		LS	EX	
iter_k	EX		MP	СМ	P2		LR	EX		MP	СМ	P2		LS	EX	
iter_avg	EX		MP	CM"	P2		LR	EX		MP	СМ	P2		LS	EX	
avgWave	EX		MP	CM	P2		LR	EX		MP	СМ	P2		LS	EX	
haarWave	EX		MP	СМ	P2		LR	EX		MP	СМ	P2		LS	EX	

Figure 8: KOJAK Performance Trends for 1to1r_1024 for Each Method at Default Thresholds.

Indeed, the wavelet transforms produced slightly larger files for the benchmark traces; however, the reduced traces of sweep3d were smaller than those produced by *Euclidean*.

To determine best method for comparing traces, we take the highest ranking methods from each category and weigh the importance of each of the categories. The best methods from the size category were iter_avg, followed by avgWave, haarWave, and Chebyshev. Those from the approximation distance category were relDiff and absDiff, followed by iter_avg. Finally, the methods that best retained performance trends were avgWave, Manhattan, Euclidean, and haarWave. One could argue that the absolute most important criteria for judging these methods is whether or not they retain the correct performance trends, because that is the point of collecting the traces in the first place. However, almost equally important is the ability to collect, store, and analyze the trace data at all. Given that avgWave performed well in both the size and retention of performance trends categories, we choose avgWave as the best method of the ones studied for comparing traces.

6. CONCLUSIONS

We have developed a new methodology for evaluating definitions for similarity between event traces for the purpose of performance analysis. We identified criteria for comparing the

REFERENCES

- [1] The ASCI sweep3D readme file. http://www.c3.lanl.gov/pal/software/sweep3d/sweep3d_readme. html, January 2009.
- [2] M.G. Aguilera, P.J. Teller, M. Taufer, and F. Wolf. A systematic multi-step methodology for performance analysis of communication traces of distributed applications based on hierarchical clustering. In *IPDPS*, 2006.
- [3] L. Carrington, A. Snavely, X. Gao, and N. Wolter. A performance prediction framework for scientific applications. In *Workshop on Performance Modeling ICCS*, 2003.
- [4] M. Casas, R. M. Badia, and J. Labarta. Automatic phase detection of MPI applications. In C. H. Bischof, H. M. Bücker, P. Gibbon, G. R. Joubert, T. Lippert, B. Mohr, and F. J. Peters,

similarity methods: file size reduction, degree of matching, approximation distance, and retention of correct performance trends. We applied these criteria, using benchmarks with known performance behaviors, as well as with the application sweep3d. Overall, the *avgWave* method had the best retention of performance behaviors and good trace file size reduction. The greatest trace file reductions were achieved with the *iter_avg* method; however, the error in those traces led to loss of important performance trends in the data. Because of this we found that using the *avgWave* method was the best trade-off in terms of error in the reduced trace and file size reduction.

Future directions for this work include investigating additional difference methods, such as trace sampling; and evaluating the methods against a richer set of full application traces.

7. ACKNOWLEDGMENTS

We thank our reviewers for their valuable suggestions; and Bernd Mohr and Felix Wolf for feedback on an earlier draft. This work was supported in part by the U.S. Department of Energy's Office of Science through the SciDAC2 award entitled Performance Engineering Research Institute, and by a Portland State University Faculty Enhancement Grant. We thank Lawrence Livermore National Laboratory for machine cycles and storage resources.

- editors, *PARCO*, volume 15 of *Advances in Parallel Computing*, pages 129–136. IOS Press, 2007.
- [5] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *Data Engineering*, 1999. Proceedings., 15th International Conference on, pages 126–133, Mar 1999.
- [6] I.-H. Chung, R. E. Walkup, H.-F. Wen, and H. Yu. MPI performance analysis tools on Blue Gene/L. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing (SC'06)*, page 123, New York, NY, USA, 2006. ACM.
- [7] T. Fahringer, M. Gerndt, B. Mohr, F. Wolf, G. Riley, and J. Traff. Knowledge specification for automatic performance analysis. Technical Report Revised Edition, ESPRIT IV Working Group on Automatic Performance Analysis: Resources and Tools (APART), http://www.fz-juelich.de/apart-1/reports/wp2-asl.ps.gz, January 2001.

- [8] F. Freitag, J. Corbalan, and J. Labarta. A dynamic periodicity detector: Application to speedup computation. In *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01), San Francisco, CA, USA*, April 23-17 2001.
- [9] T. Gamblin, B. de Supinski, M. Schulz, R. Fowler, and D. Reed. Scalable load-balance measurement for SPMD codes. In SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, 2008.
- [10] T. Gamblin, R. Fowler, and D. A. Reed. Scalable methods for monitoring and detecting behavioral equivalence classes in scientific codes. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'08), Miami, FL*, April 14-28 2008.
- [11] M. Gerndt, B. Mohr, and J. L. Träff. A test suite for parallel performance analysis tools. *Concurrency and Computation: Practice and Experience*, 19(11):1465–1480, August 2007.
- [12] W. Gu, G. Eisenhauer, K. Schwan, and J. Vetter. Falcon: on-line monitoring and steering of large-scale parallel programs. *Concurrency: Practice and Experience*, 10(9):699–736, Dec 1998
- [13] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2005.
- [14] M. Hauswirth, A. Diwan, P. F. Sweeny, and M. C. Mozer. Automating vertical profiling. In *Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications*, pages 281 296, October 16-20 2005.
- [15] J. Hollingsworth, B. Miller, and J. Cargille. Dynamic program instrumentation for scalable performance tools. In *Proceedings of Scalable High Performance Computing Conference, Knoxville, TN, USA*, pages 841–850, May 23-25 1994.
- [16] T. Huffmire and T. Sherwood. Wavelet-based phase classification. In *PACT '06: Proceedings of the 15th international conference on Parallel architectures and compilation techniques*, pages 95–104, New York, NY, USA, 2006. ACM.
- [17] A. Jensen and A. la Cour-Harbo. *Ripples in Mathematics: The Discrete Wavelet Transform*. Springer-Verlag, 2001.
- [18] L. V. Kalé, S. Kumar, G. Zheng, and C. W. Lee. Scaling molecular dynamics to 3000 processors with projections: A performance analysis case study. In *International Conference on Computational Science (ICCS 2003)*, *Melbourne*, *Australia and St. Petersburg*, *Russia*, pages 23–32, June 2-4 2003.
- [19] A. Knüpfer. A new data compression technique for event based program traces. In *International Conference on Computational Science*, pages 956–965, 2003.
- [20] D. Kranzlmüller, S. Grabner, and J. Volkert. Event graph visualization for debugging large applications. In SPDT '96: Proceedings of the SIGMETRICS symposium on Parallel and distributed tools, Philadelphia, Pennsylvania, USA, pages 108–117, 1996.
- [21] D. Kranzlmüller, A. Knüpfer, and W. E. Nagel. Pattern matching of collective MPI operations. In *Proceedings of the International Conference on Parallel and Distributed*

- Processing Techniques and Applications (PDPTA '04), Las Vegas, Nevada, USA, June 21-24 2004.
- [22] D.B. Lange and Y. Nakamura. Object-oriented program tracing and visualization. *IEEE Computer*, 30(5):63–70, May 1997.
- [23] C. W. Lee, C. Mendes, and L. V. Kalé. Towards scalable performance analysis and visualization through data reduction. In 13th International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS 2008) held in conjunction with IPDPS 2008, 2008.
- [24] K. Mohror and K. L. Karavanic. Towards scalable event tracing for high-end systems. In *High Performance Computing and Communications, Third International Conference (HPCC 2007), Houston, Texas, USA*, pages 695–706, September 26-28 2007.
- [25] O. Nickolayev, P. Roth, and D. Reed. Real-time statistical clustering for event trace reduction. *International Journal of High Performance Computing Applications*, 11(2):69–80, 1997.
- [26] M. Noeth, F. Mueller, M. Schulz, and B. R. de Supinski. Scalable compression and replay of communication traces in massively parallel environments. In 21th International Parallel and Distributed Processing Symposium (IPDPS'07), March 2007
- [27] F. Petrini, D. J. Kerbyson, and S. Pakin. The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of ASCI Q. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing (SC'03)*, *Phoenix, Arizona, USA*, page 55, November 15-21 2003.
- [28] P. Ratn, F. Mueller, B. R. de Supinski, and M. Schulz. Preserving time in large-scale communication traces. In *ICS '08: Proceedings of the 22nd annual international conference on Supercomputing*, pages 46–55, New York, NY, USA, 2008. ACM.
- [29] F. Song, F. Wolf, N. Bhatia, J. Dongarra, and S. Moore. An algebra for cross-experiment performance analysis. In *Proc. of the International Conference on Parallel Processing (ICPP)*, pages 63–72, Montreal, Canada, August 2004. IEEE Society.
- [30] D.P. Spooner and D.J. Kerbyson. Performance feature identification by comparative trace analysis. *Future Generation Comp. Syst.*, 22(3):369–380, 2006.
- [31] E. Thereska, B. Salmon, J. D. Strunk, M. Wachs, M. Abd-El-Malek, J. Lopez, and G. R. Ganger. Stardust: tracking activity in a distributed storage system. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS 2006, Saint Malo, France*, pages 3–14, June 26-30 2006.
- [32] J. Vetter. Dynamic statistical profiling of communication activity in distributed applications. In *Proceedings of ACM SIGMETRICS 2002 International Conference on Measurement and Modeling of Computer Systems, Marina Del Rey, CA, USA*, pages 240–250, June 15-19 2002.
- [33] J. S. Walker. A Primer on Wavelets and Their Scientific Applications. Chapman & Hall/CRC, 2008.
- [34] F. Wolf, B. Mohr, J. Dongarra, and S. Moore. Automatic analysis of inefficiency patterns in parallel applications. *Concurrency and Computation: Practice and Experience*, 19:1481–1496, 2007.

- [35] J. C. Yan, H. H. Jin, and M. A. Schmidt. Performance data gathering and representation from fixed-size satistical data. Technical Report NAS-98-003, NASA Ames Research Center, 1908
- [36] J. C. Yan and M. Schmidt. Constructing space-time views from fixed size trace files getting the best of both worlds. In

Parallel Computing: Fundamentals, Applications and New Directions, Proceedings of the Conference (ParCo'97), Bonn, Germany, pages 633–640, September 19-22 1997.

APPENDIX

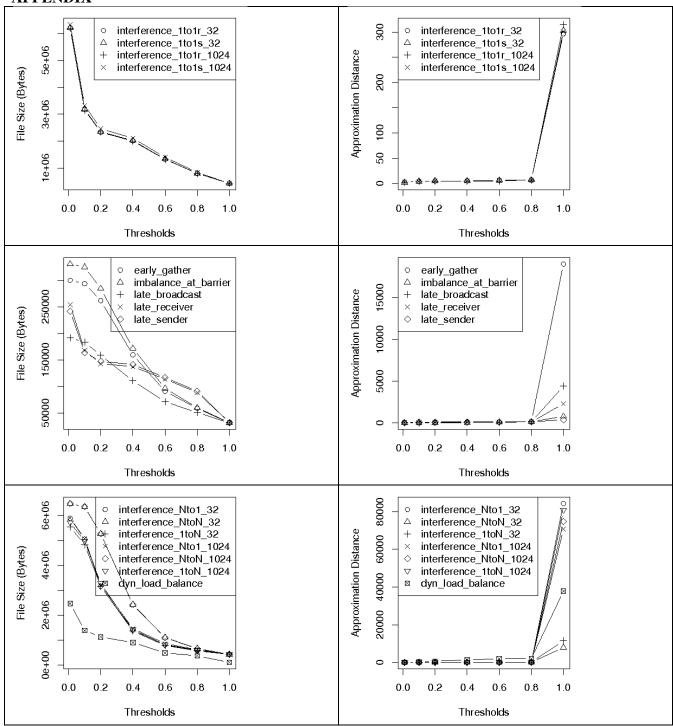


Figure 9: File Size and Approximation Distance for Varying Duration Thresholds and Relative Distance

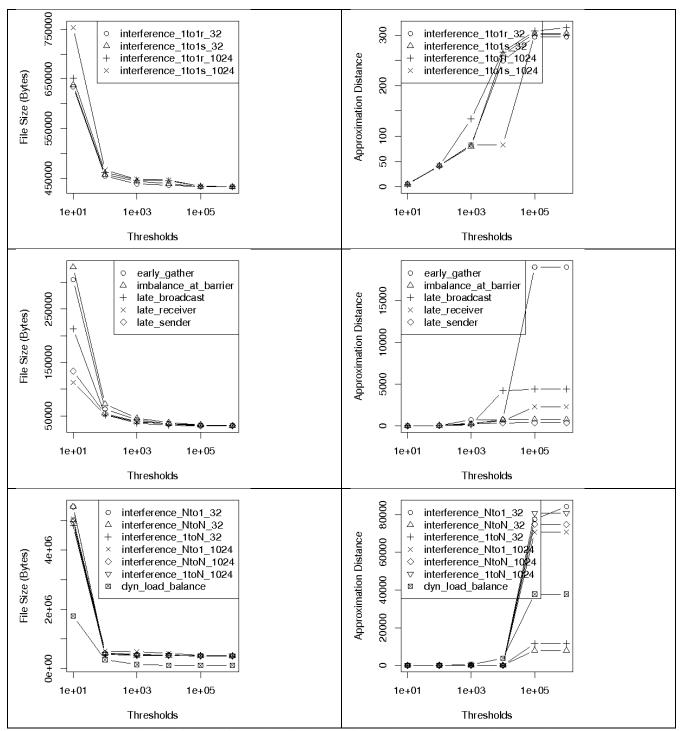


Figure 10: File Size and Approximation Distance for Varying Threshold and Absolute Distance

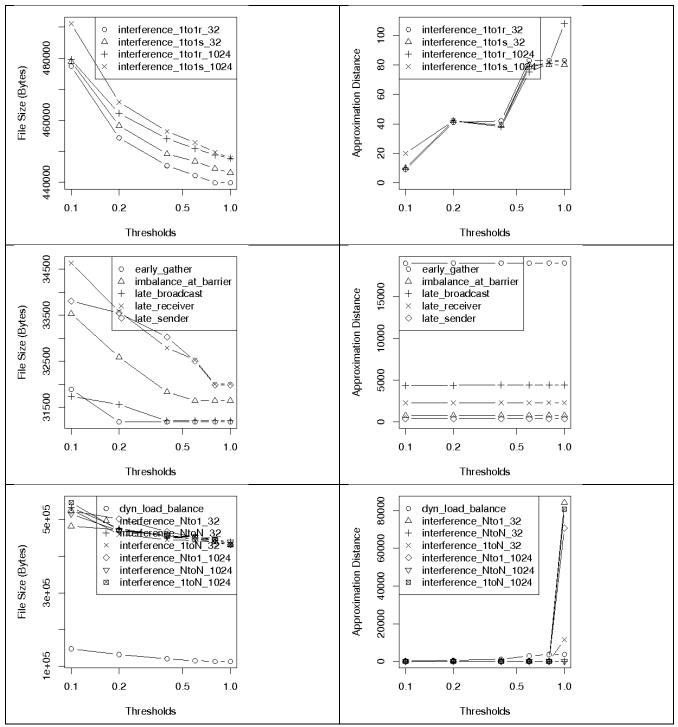


Figure 11: File Size and Approximation Distance for Varying Threshold and Manhattan Distance

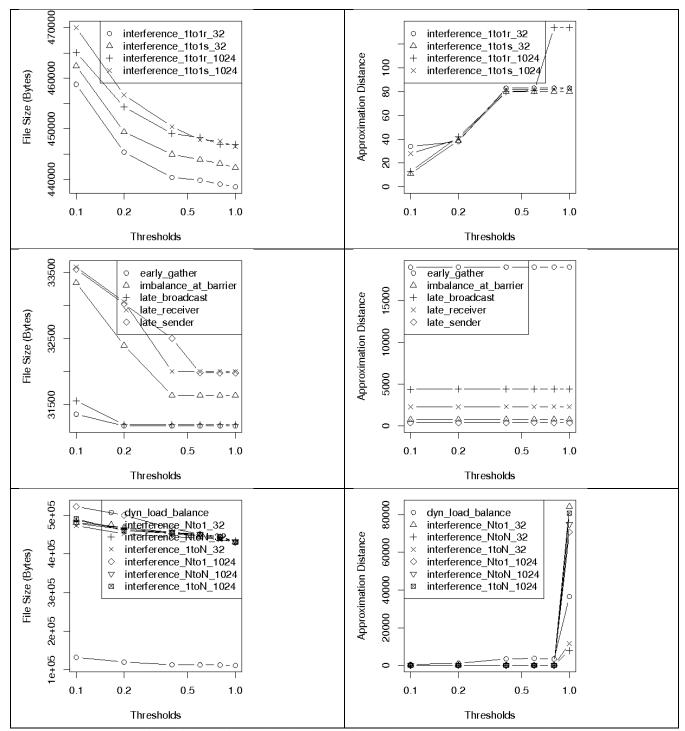


Figure 12: File Size and Approximation Distance for Varying Threshold and Euclidean Distance

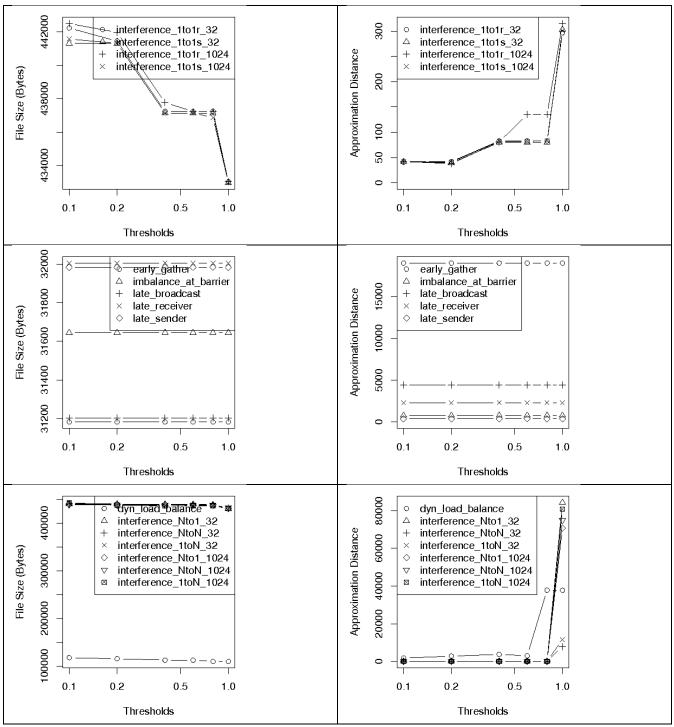


Figure 13: File Size and Approximation Distance for Varying Threshold and Chebyshev Distance

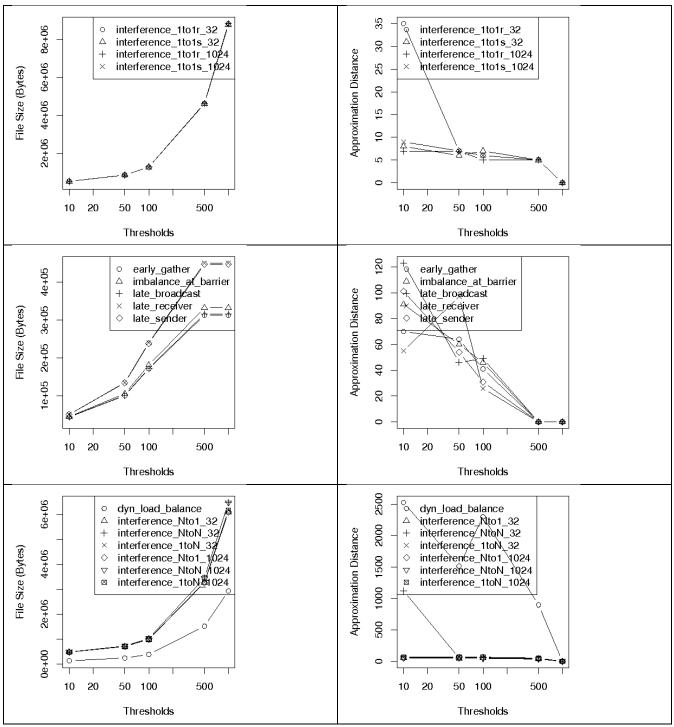


Figure 14: File Size and Approximation Distance for Varying Threshold and Keep k Iterations

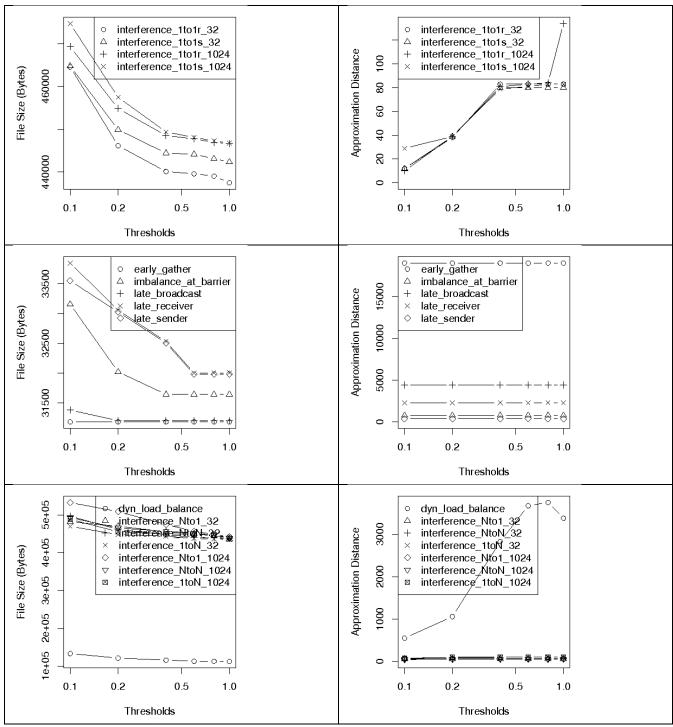


Figure 15: File Size and Approximation Distance for Varying Threshold and Average Wavelet Transform

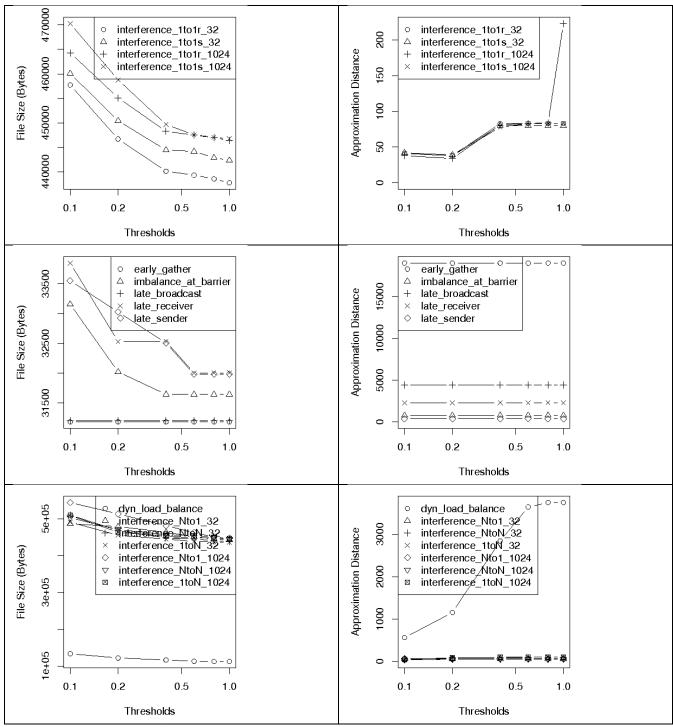


Figure 16: File Size and Approximation Distance for Varying Threshold and Haar Wavelet Transform

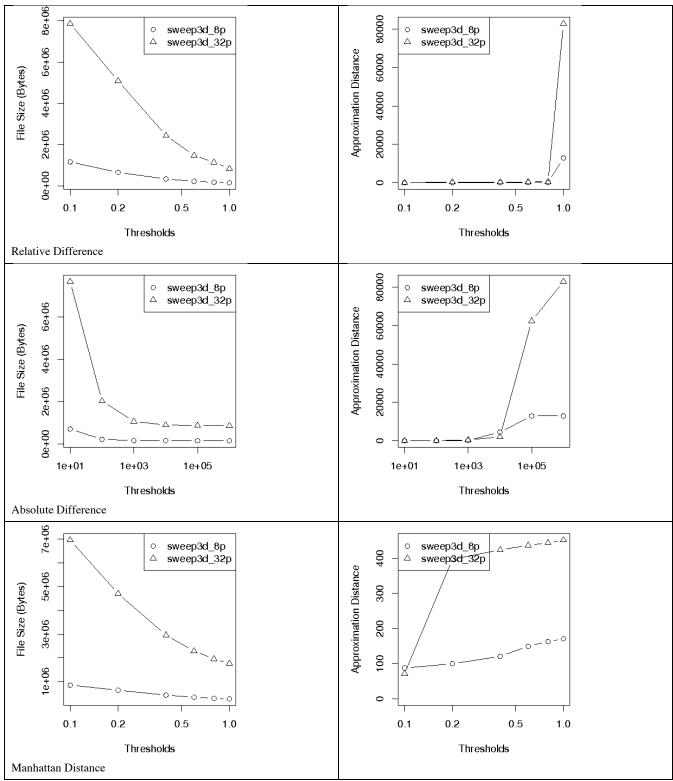


Figure 17: File Size and Approximation Distance for Varying Thresholds for Sweep3d and relDiff, absDiff, Manhattan

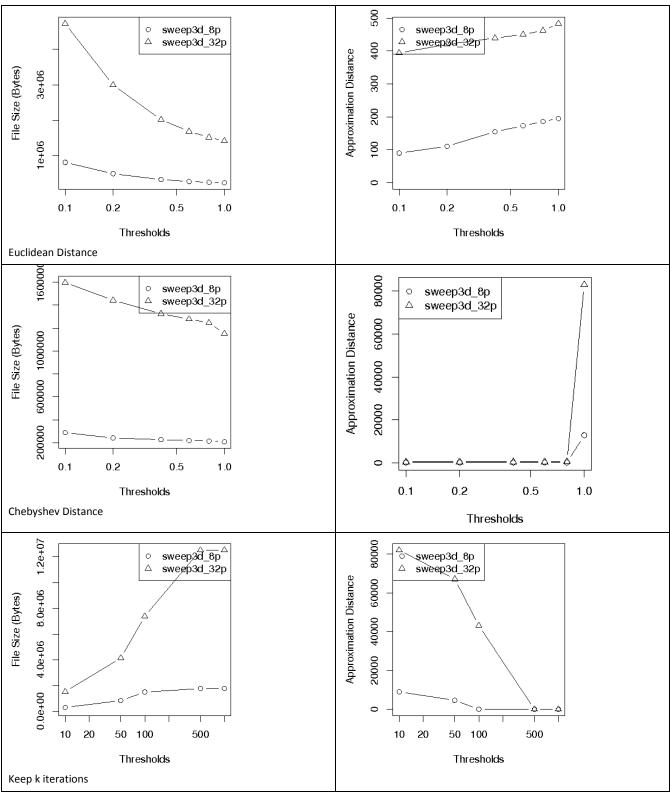


Figure 18: File Size and Approximation Distance for Varying Thresholds for Sweep3d and Euclidean, Chebyshev, iter_k

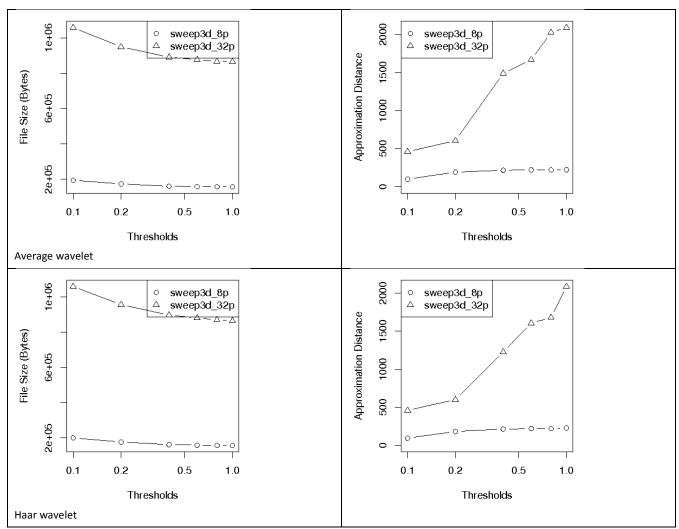


Figure 19: File Size and Approximation Distance for Varying Thresholds for Sweep3d and Wavelet Transforms

Table 1: Retention of Performance Trends with Varying Thresholds for dyn_load_balance

	1		MPI_Alltoall							
	no loss	EX	MP	CM	0.0	NN STATE OF THE ST	do_work			
	0.1	Charles	MPC	C/A	EØ.	NN III	EX			
	0.2	EX.	SAP	CM	60	NN MINE	EX			
به	0.4	E & CHICAGO BY	PAP CONTRACTOR	EMBINE.	100	NN	EX			
relative	0.6	5,32	to the same of the same	CN	COL	NN	EX			
relative	0.8	5.5 (100)	BOT STORES	EW SHIP	CÓ	NN	EX			
한 등	1.0	EX .	MP	EM	CO	NN	EX-			
	10	EX COMMENTS	MP	CM	(3)	NN MARKET	EX			
	100	UK COMPANY	saf programme	CV	00	NN III	EX			
44	1000		SAF	£M.	(0)	NN B	EX			
absolute	10000	ex	MAP	EM	(0)	NN	EX			
absolute differenc	100000	EX EXECUTION OF THE PROPERTY O	MP	CM	con	NN	EX			
de de	100000	EX	MP	CM	CO	NN	EX			
59 191	0.1	EX M	ME	CM	CO		CX			
		I.A.	ORF CONTRACTOR	2.00	CU CU	NN	EX			
_	0.2	E.F.	WA DE	a ca		NN B	EX			
e a	0.4	1.1	W IIII	LW		NN Market Ball	EX			
Manhattan distance	0.6	Ex manual	MP	- EM		NN	EX			
/ar	0.8	EX	MP	CM	CO	NN	EX			
2 0	1.0	EX MINISTER	BAP	EM EM	CO	NN	EX			
	0.1	EX	MP	CM	60	NN I	EX			
	0.2	U.A. (Comp.)	MP (III)	EN PHONE	1.6	NN NN	EX			
= 00	0.4	EX manufacture	MAP	CW	CO	NN NN	EX			
nce ac	0.6	EX	MP	CM	CO	NN	EX			
Euclidean distance	0.8	EA MITTER	MP	CM	CO	NN	EX			
E . E	1.0	EX CONTRACTOR	MP	CM	CO	NN III	EX			
	0.1	EX.	MP E	EM .	60	NN E	EX			
	0.2		MP	0.00	0.0	NN NN	EX			
>	0.4	EX	net e	CW.	CO	NN	EX			
She She	0.6	15	0.0		(6)	NN	EX			
Chebyshev	0.8	EX EX	MP	CM	co	NN	ex			
£ 5	1.0	EX	MP	CM	CO	NN	EX			
	0.1	EA .	200	CH	50	NN B	EV			
	0.2	EX.	And the same of	CM CM	- 25	NN COLUMN	EV III			
	0.4	EX.	SAP	- CV	7.6	NN	EX			
Average	0.6	EX	042	CM	CO	NN STATE OF	11			
Average	0.8	EX	MAP	CM	CO .	NN NN	EX			
& B	1.0	EX	TAP	CM	65	NN	EX			
	0.1	I CY	SAP	CM I	CÖ	NN NN	EX			
et	0.2	13.	262	EM	1.6	NN NN	EX			
Ne Ne	0.4	EX	569	EM	CO2	NN	EX			
×	0.6	EX	MP	CM	000	NN NN	EX			
Haar Wavelet	0.8	Ex manufacture	MP	CM	65	NN NN	EX			
Ĭ	1.0	EX	MP	C.tvl	CO	NN	E S			
	500	EX.	846	EMELLER .	CO	NN THE RESERVE	EX			
72	100	- EX	MP	CM	E0	NN	EX			
ons	50	EX PROPERTY.	SPE	CM	CD)	NN CONTRACTOR	EX			
rati	10		100		100	NN	EX			
Keep k iterations	1	EX manufacture	ASD	CM	CO		EX			
	average	EX	MP	CM	CO	NN	EX.			

Table 2: Retention of Performance Trends with Varying Thresholds for early_gather

			-	MPI_Gather			do_work
	no loss	EX	MP	CM	CO	ER	EX
	0.1	EX	MP	CM	CO	ER	EX
	0.2	EX	MP	CM	CO	ER	EX
9	0.4	EX	MP	CM	CO	ER	EX
e ve	0.6	EX	MP	CM	CO	ER	EX
difference	0.8	EX	MP	CM	CO	ER	EX .
2 =	1.0	EX	MP	CM	CO .	ER	EX TOTAL
	10	EX	MP	CM	CO	ER	EX
	100	EX	MP	CM	CO	ER	EX CONTRACTOR
9)	1000	EX	MP	CM	COM	ER	EX
en de	10000	EX	MP	CM	CO	ER	EX CONTRACTOR
absolute difference	100000	EX	MP	CM	CO	ER I	EX
9 9	1000000	EX	MP	CM	CO	ER	EX .
	0.1	EX THE RES	MP	CM	CO	ER 🔳	EX
	0.2	EX CONTRACTOR	MP	CM	CO	ER I	EX
E .	0.4	EX	MP	CM	COBB	ER	EX
Manhattan distance	0.6	EX	MP	CM	CO	ER	EX
tan	0.8	EX .	MP	CM	COBB	ER	EX
dis A	1.0	EX	MP	CM	CO	ER	EX
	_			1			
	0.1	EX	MP	CM	CO	ER	EX
	0.2	EX W	MP	CM	CO	ER	EX
e au	0.4	EX	MP	CM III	CO	ER	EX
distance	0.6	EX	MP	CM	CO	ER	EX
distance	0.8	EX	MP	CM B B B	COMM	ER	EX
	1.0	EX	MP	CM	CO E	ER 📰	EX
	0.1	EX	MP	CM	CO	ER 📕	EX
	0.2	EX	MP	CM B	CO	ER 📰	EX
e e	0.4	EX Manual Base	MP	CM	CO	ER	EX
Chebyshev distance	0.6	EX	MP	CM III	CO	ER	EX
chebyshe	0.8	EX	MP	CM	CO	ER	EX ESTATEMENT
J 0	1.0	EX W	MP	CM	CO	ER 🔳	EX E
	0.1	EX	MP	CM	CO	ER 🔳	EX TOTAL
	0.2	EX	MP	CM	COMM	ER 🔳	EX CONTRACTOR
u =	0.4	EX	MP	CM	CO	ER I	EX CONTRACTOR
vel	0.6	EX	MP B	CM	COM	ER I	EX
Average	0.8	EX	MP	CM	CO	ER	EX
	1.0	EX	MP B	CM III III	COM M	ER	EX
<u></u>	0.1	EX	MP	CM	CO	ER	EX
ndar wavelet	0.2	EX	MP	CM		ER	EX
S	0.4	EX EX	MP	CM		ER III	EX
0	0.8	EX EX	MP MP	CM CM	COMM	ER E	EX EX
	1.0	EX	MP	CM		ER m	EX
	500		MP	CM			EX
	100	EX		-		ER	
SIIIS		EX	MP	CM		ER	EX
atic	50	EX	MP	CM		ER	EX CONTRACTOR
Keep k iterations	10	EX	MP	CM		ER	EX EX
100000	1	EX	MP	CM III	CON I	ER	EX

 $Table \ 3: Retention \ of \ Performance \ Trends \ with \ Varying \ Threshold \ for \ imbalance_at_mpi_barrier$

			du -	MPI_Barrier			do_work
	no loss	EX CONTROL	MP	SN CONTRACTOR	SA	WESTER	EX
	0.1	EX	MP	SN	8A	WROME	EX
	0.2	EX CONTRACTOR	MP	SN	8A	WE	EX
Q.	0.4	EX CONTRACTOR	MP	SN	BA BA	WB	EX
difference	0.6	EX CONTRACT	MP	SN ELECTRON	8A	WB	EX
differen	0.8	EX	MP	SN	BA BA	WB	EX
=======================================	1.0	EX CONTRACTOR	MP	SN	8A	WB	EX
	10	EX	MP	SN	8A	West	EX
	100	EX	MP	5N	8A	WB	EX
23	1000	EX TOTAL	MP	SN S	8A	WE STATE OF THE ST	EX
difference	10000		MP	SN		wes	EX
differenc							
H	100000	EX	MP	SN CHARLES	8A	WB	EX
0	1000000	EX	MP	SN	BA	We	EX
	0.1	EX	MP	SN CONTRACTOR	BA	Wegine	EX
	0.2	EX	MP	SN	8A	W. G. STREET,	EX
a)	0.4	EX CONTRACTOR	MP	SN SN	BA E	WB	EX
distance	0.6	EX manufacture	MP	SM CONTRACTOR	8A	WE	EX
ista	0.8	EX COMMENTS	MP	5N CONTRACTOR	EA AS	WB (EX
-	1.0	EX	MP	SN	A.S	WE	EX
	0.1	EX (IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	MP MINE	SN CONTRACTOR	BA MANAGEMENT	WB	EX CONTRACTOR
distance	0.2	EX CONTRACTOR	MP	SN	EA AS	W6 (Control of the Control of the Co	EX
	0.4	EX CONTRACTOR	MP	SN STATE OF THE ST	8A	WB COMMENTS	EX
	0.6	EX CONTRACTOR	MP	SN CONTRACTOR	8A	WREST	EX
distance	0.8	EX	MP	5N	8A	WEST	EX
-6	1.0	EX	MP	SN CONTRACTOR	SA CONTRACTOR	we	EX
	0.1	EX	MP	SN	BA CONTRACTOR	WB	EX
	0.2	EX CONTRACTOR	MP	SN CONTRACTOR OF THE STATE OF T	8A	Wegleson	EX
	0.4	Ex manufacture	MP	SN	AS AS	West	EX
distance	0.6	EX	MP	5N	8A	WB	EX
distance	0.8			SN		12.00	EX
dis	1.0		Maria de la companya della companya della companya della companya de la companya della companya				
8	The same of the sa	EX	MP	SN MILES	AS ASSESSMENT AS	WB	EX
	0.1	EX	MP	SN	8.A	WE	EX
	0.2	EX	MP	SN	8A 8A	88W	EX
<u> =</u>	0.4	EX CONTRACTOR	MP MP	SN SN	BA BA	WB	EX
Wavelet	0.8	EX	MP	SN S	&A A	WB	EX
3	1.0	EX	MP	SN CONTRACTOR	8A	WB	EX
	- American	1	MO	1	PA .	W/P	
	0.1	EX EX	MP	SN SN	8A	WB	EX EX
	0.4	EX	MP	SN CONTRACTOR	8A	WB	EX
	0.6	EX	MP	SN S	6A	WB	EX
	0.8	EX	MP	SN	8A	WB	EX
	1.0	EX	MP	5N	BA	WB	EX
	500	EX	MP	SN SN	8A	WB	EX
- 83	100	EX	MP	SN	EA .	WB	EX
iterations	50	EX MANUAL	MP	SN CONTRACTOR	BA BA	we	EX
iteratio	1				and the second second	COLUMN TO THE REAL PROPERTY.	EX
e.	10		MP	+			
	1	EX CONTRACTOR	MP	5N SN	BA BA	WE THE RESERVE	EX EX

Table 4: Retention of Performance Trends with Varying Threshold for late_broadcast

	il.			MPI_Bca	ist		do_work
	no loss	EX III	MPE	CME	COS	LB III	EX
	0.1	EX III	MP	CM	COE	LB III	EX
	0.2	EX II	МРШ	CM	COM	LBU	EX
a.	0.4	EXT	MPE	CME	CO	LB fil	EX
en c	0.6	EX III	MPE	CM	COM	LBE	EX
difference	0.8	EXID	MP	СМ	co	LER	EX
5 5	1.0	EX	MP	CME	CO	LB E	EX
	10	EX III	MPW	CMII	COM	E8 III	EX
	100	EX II	мене	CMIE	COL	LB	EX
	1000	EX DE	MPINE	CME	COS	1.8 11	EX
en c	10000	EX II	MP	CM	CO	LB E	EX EX
difference	100000	EX III	MPDE	СМЕ	COM	18 6	EX
a	1000000	EX III	MP	CMBE	COM	LB	EX
	0.1	EX III	MP	CMI	COS	LB	EX
	0.2	EX III	MPI	CMI	CO	18.0	EX
	0.4	EX DE	мене	CM	CO.	LER	EX
distance	0.6	EXI	MP	CM	CO	18.00	EX
distance	0.8	EX III	MPE	CME	COM	1.8.83	EX
dis	1.0	EX III	MP	EM	con	1.0	EX
-17-	0.1	EX	MP	CME	COM	LB	EX
	0.2	EX III	MPIFE	CMIE	COS	10.0	EX
	0.4	EX D	MPULE	CMILE	CO	LP C	EX
e	0.6	EX E	MPIE	CME	50	1.0	EX
distance	100401	EX II	MPE	CMEE	(3)	1.0	EX
distance	0.8	EX III			COR	1.0	EX
	100000		MP	CM		1.0	
	0.1	EX III	MPRE	CMEE	COM	100	EX EX
			MP	CMI	COM	1.0	
9	0.4	EX III	MP	CM	COM	1.0	EX
distance	50000	EX III	MPIE	CMIE	COM	1.0	EX
distance	0.8	EX III	MP	EME	COM	LB III	EX
	1.0	EX II	MP	CM	CO	CR S	EX .
	0.1	EX III	MP	CM	COM	T LB /II	EX
	0.4	EX III	MP	CM	60	LB S	EX EX
<u> </u>	0.6	EX I	MP MP	CM	COM	12 (EX
Wavelet	0.8	EX II	MPIBE	CME	(0)	1.8 10	EX
3	1.0	EX III	MPRO	CM	COS	LB =	EX
	0.1	EXI	MP	CME	COM	LER	EX
	0.2	EX III	MPO	CM	CON	LB II	EX
	0.4	EX H	naPage	CMI	CO	LB #	EX
	0.6	EX III	MPIN	CME	001	LB	EX
	0.8	EX III	MPB	CME	CO#	LB #	EX .
100	1.0	EX III	MPE	СМа	COR	LB III	EX
	500	EX III	MPD	СМЕ	COM	LB III	EX
100	100	EXII	MP	CME	CO	LB III	EX
4 .0	50	EX III	MP	CM	COM	LB E	EX EX
iterations	10	EX III	0.461 (0.00)	CM	COM	LB	EX EX
E . E	1	EX III	MPIN	CM	COS	LB III	EX
	average	EXID	MPER	CM	COM	18.6	EX MANAGEMENT

Table 5: Retention and Performance Trends with Varying Thresholds for late_receiver

				MPI_Ssend			do_work
	no loss	EX CONTRACTOR	MP	CM	P2 3 3 3 3	LR COMMENT	EX
	0.1	EX MINISTRA	MP	CM	P2 11 11 11 11	LR	EX
	0.2	EX	MP	CM	P2	LR	EX
e e	0.4	EX MINISTRA	MP	CM	P2	LR	EX
ë	0.6	EX WIND W	MP	CM	P2	LR	EX
difference	0.8	EX	MP	CM	P2	LR	EX
-5	1.0	EX	MP	CM	P2	LR	EX
	10	EX .	MP	CM	P2	LR	EX
	100	EX .	MP	CM	P2	LR	EX
e e	1000	EX .	MP	CM	P2	LR	EX .
ence	10000	EX E	MP	CM	P2	LR	EX CONTRACTOR
difference	100000	EX CONTRACTOR	MP	CM	P2	LR	EX CONTRACTOR
ē	1000000	EX BERRETA	MP	CM	P2	LR	EX III
	0.1	EX THE RESERVE	MP	CMBBBBB	P2	LR	EX
	0.2	EX MINIMUM	MP	CM	P2 	LR	EX
	0.4	EX	MP	CM	P2	LR	EX
distance	0.6	EX COLOR	Toronto Maria Control Control	CM	P2	LR	EX
distance	0.8	The second secon	MP			7.75	EX
dis	1.0	EX WIND TO SERVICE OF THE SERVICE OF	MP	CM	P2	LR	
		EX TOTAL	MP	CM	P2	LR	EX
	0.1	EX	MP	CM	P2	LR	EX
	0.2	EX	MP	CM	P2	LR	EX
Luclidean	0.4	EX III III III	MP B B B	CM	P2 11 11 11 11	LR	EX
ğ	0.6	EX	MP	CM	P2	LR	EX
distance	0.8	EX	MP	CM	P2	LR	EX
1.0	1.0	EX MINISTRA	MP M M	CM	P2	LR	EX
	0.1	EX	MP	CM	P2	LR	EX CONTRACTOR
	0.2	EX	MP	CM	P2	LR	EX CONTRACTOR
200	0.4	EX M M M	MP	CM	P2	LR	EX
distance	0.6	EX	MP	CM	P2	LR	EX
sta	0.8	EX THE REST	MP	CM	P2	LR	EX
. 0	1.0	EX WEST	MP	CM	P2	LR	EX .
	0.1	EX	MP	CM	P2	LR	EX
	0.2	EX .	MP	CM	P2	LR	EX .
	0.4	EX .	MP	CM	P2	LR	EX
Wavelet	0.6	EX MINISTER	MP	CME III	P2 11 11 11 11	LR	EX
Nav	0.8	EX W	MP	CM	P2	LR	EX .
. >	1.0	EX	MP	CM	P2	LR	EX
	0.1	EX	MP	CM	P2	LR	EX .
	0.2	EX	MP M M M	CM	P2	LR	EX CONTRACTOR
	0.4	EX M M M	MP	CM	P2	LR	EX
	0.6	EX MINING	MP	CM	P2	LR	EX STATE OF THE ST
	0.8	EX WINDS	MP	CM	P2	LR	EX :
	1.0	EX	мР	CM	P2	LR	EX
	500	EX	MP	CM	P2	LR MINISTER	EX
NS.	100	EX X	MP	CMD	P2	LR	EX
tio	50	EX	MP	CM	P2	LR	EX CONTRACTOR
iterations	10	EX MIN MIN	MP	CM	P2	LR	EX
	1	EX	MP	CMB B B	P2	LR	EX
	average	EX	MP	CM			

Table 6: Retention of Performance Trends with Varying Thresholds for late_sender

				do_work			
	no loss	EX B B B	MP	CM	P2	LS	EX
	0.1	EX TO THE TOTAL TO	MP	CM	P2	LS	EX CONTRACTOR
	0.2	EX	MPI	CM B B B	P2	LS	EX .
9	0.4	EX	MP	CM	P2	LS	EX EX
en en	0.6	EX TOTAL NAME OF THE PARTY OF T	MP	CM	P2	LS	EX .
relative	0.8	EX	MP	CM B B B	P2	L5	EX
2 -	1.0	EX	MP	CM	P2	1.5	EX III
	10	EX III III III III	MP	CM	P2	LS	EX
	100	EX IIII III III	MP	CM B B B	P2	LS	EX CONTRACTOR
. 8	1000	EX	MP	CM	P2	LS	EX EXECUTED IN
ence	10000	EX III III III	MP	CM	P2	L5	EX CONTRACTOR
absolute difference	100000	EX TOTAL DE	MP	CM	P2	LS	EX
T 0	1000000	EX E	MP .	CM	P2	LS	EX .
	0.1	EX B B B	MP	CM	P2	LS	EX
	0.2	EX	MP	CM	P2	LS	EX CONTRACTOR
an	0.4	EX	MP	CM	P2	LS	EX
Manhattan distance	0.6	EX B B B	MP	CM	P2	LS	EX
star	0.8	EX IIII	MP	CM	P2	LS	EX
≥ 5	1.0	EX .	MP	CM	P2	LS DECEMBER	EX TOTAL
	0.1	EX	MP B B B	CM	P2	LS	EX
	0.2	EX IIII	MP	CM B B B	P2	LS III	EX
_	0.4	EX	MP	CM B B B	P2	LS	EX CONTRACTOR
Euclidean distance	0.6	EX B B B	MP	CM	P2	LS	EX
Euclidea	0.8	EX BOOK B	MP	CM CM	P2	LS	EX
dis G	1.0	EX B B B	MP	CM B B B	P2	LS	EX
	0.1	EX BBB	MP .	CM	P2	LS	EX
	0.2	EX B D D	MP	CM CM	P2	LS	EX CONTRACTOR
>	0.4	EX	MP	CM	P2	LS	EX
Chebyshev distance	0.6	EX B B B	MP	CM	P2	LS	EX
tan	0.8	EX BBB	MP	CM	P2	LS	EX
£ :8	1.0	EX BBBB	MP	CM B B B	P2	LS	EX
	0.1	EX B B B	MP	CM	P2	LS	EX
	0.2	EX B B B	MP	CM	P2	LS	EX
	0.4	EX B B B	MP	CM B B B	P2	LS BBBBB	EX
age	0.6	EX	MP	CM	P2	LS	EX
Average	0.8	EX	MPHERE	CM	P2	LS	EX
K S	1.0	EX THE REST	MP	CM BE B	P2	LS IIII	EX .
	0.1	EX D	MP	CM		LS DECEMBER 1	EX
let	0.2	EX III III III III	MP	CM B B B	P2	LS III	EX CONTRACTOR
ave	0.4	EX EX EX	MP	CM B B B	P2	LS M M M M	EX
Haar Wavelet	0.6	EX EX EX	MPERSON	CM	P2	LS IIII	EX E
Lag	0.8	EX WWW	MP	CM	P2	LS	EX E
_	1.0	EX	мРамента	CM	P2	LS III	EX
	500	EX B B B	MP	CM B B B	P2	LS	EX
SI	100	EX	MP	CM	P2	LS	EX
p k	50	EX	MP	CM	P2	LS	EX CONTRACTOR
Keep k iterations	10	EX IIIII	MP	CM	P2	LS	EX
× .=	1	EX B B B	MP B B B	CM	P2	L5	EX CONTRACTOR
	average	EX B B B	MP	CM	P2	LS	EX

Table 7: Retention of Performance Trends with Varying Thresholds for Nto1_32

	T.			MPI_Gathe	r		do_work
	no loss	EX III	MP	CM	CO	ER	EX 📉
	0.1	EX	MP	CM	CO	ER	EX E
	0.2	EX III	MP	CM	CO	ER .	EX 🔤
a.	0.4	EX III	MP	CM	CO	ER	EX E
e :	0.6	EX	MP	CM	CO	ER .	EX
difference	0.8	EX I	MP	CM	co	ER .	EX
. 6	1.0	EX III	MP	CM	CO	ER .	EX
	10	EX III	MP	CM	CO	ER	EX
	100	EX	MP	CM	co	ER	EX
	1000	EX EX	MP	CM	CO	ER	EX E
eno	10000	EX	MP	CM	co	ER	EX E
difference	100000	EX SU	MP BO	CME	CORT	ER	EX
=	1000000	EX EX	MP	CM	CO	ER .	EX
	0.1	EX	MP	CM	co	ER	EX 🔤
	0.2	EX	MP	CM	co	ER	EX E
	0.4	EX	MP	CM	CO	ER	EX E
distance	0.6		MP	CM	CO		EX EX
distance	0.8	EX				ER	EX UST
dis	1.0	EX	MP	CM	CO	ER	
2051		EX	MP	CM	CO	ER	EX
	0.1	EX	MP	CM	CO	ER	EX
	0,2	EX	MP	CM	CO	ER	EX
distance	0.4	EX E	MP	CM	CO	ER	EX
	0.6	EX III	MP	CM	CO	ER	EX E
ist	0.8	EX RES	MP	CM	CO	ER	EX USH
	1.0	EX	MP	CM	CO	ER	EX
	0.1	EX	MP	CM	CO	ER	EX
	0.2	EX III	MP	CM	CO	ER .	EX E
10000	0.4	EX I	MP	CM	CO	ER	EX E
nce	0.6	EX M	MP	CM	CO	ER	EX E
distance	0.8	EX	MP	CM	CO	ER I	EX UST
0	1.0	EX SE	MP	CM	CO	ER I	EX
	0.1	EX	MP	CM	CO	ER	EX E
	0.2	EX	MP	CM	CO	ER	EX E
-	0.4	EX	MP	CM	CO	ER	EX EX
i i	0.6	EX	MP	CM	CO	ER	EX
Wavelet	0.8	EX	MP	CM	CO	ER	EX
	1.0	EX	MP	CM	CO	ER	EX
121	0.1	EX	MP	CM	co	ER	EX 📉
	0.2	EX	MP	CM	CO	ER	EX
	0.4	EX	MP	CM	CO	ER	EX E
	0.6	EX	MP	CM	CO	ER	EX E
	0.8	EX	MP	CM	CO	ER	EX
	1.0	EX	MP	CM	CO	ER	EX
	500	EX	MP	CM	CO	ER .	EX
us	100	EX	MP	CM	CO	ER	EX
Tion I	50	EX IIII	MP	CM	CO	ER	EX
iterations	10	EX I	MP	CM	CO	ER	EX E
-	1	EX	MP	CM	CO	ER	EX
	average	EX	MP	CM	CO	ER	EX

Table 8: Retention of Performance Trends with Varying Thresholds for NtoN_32

	1			MPI	Barrier			do_work
	no loss	EX IIII	MP	SN IIII	BA IIII	WB ==	BC ==	EX
	0.1	EX IIII	MP	SN	BA	WB ME	BC -	EX
	0.2	EX IIIII	MP	SN IIII	BA	WB	BC ==	EX E
٩	0.4	EX	MP	SN III	BA III	WB	SC =	EX
difference	0.6	EX	MP	SN IIII	BA	WB	BC ==	EX
differen	0.8	EX IIII	MP	SN IIII	BA	WB.	BC ==	EX
-5	1.0	EX SSI	MP ST	SN SM	BA ST	WB	BC SON	EX IIII
	10	EX IIIII	MP	SN IIII	BA IIII	WB	BC =	EX
	100	EX IIII	MP	SN IIIII	BA	WB	BC M	EX
e)	1000	EX III	MP .	SN III.	BA .	WB -	BC -	EX
difference	10000	EX BIS	MPMM	SN ISSE	BA WE	WB	BC MIN	EX
differenc	100000	EX SI	MP W	SN SN	BA SSI	WB	BC M	EX
-5	1000000	EX S	MP M	SN SN	BA M	WB	BC MI	EX IIII
	0.1	EX IIII	MP	SN IIII	BA III	WB	BC ===	EX
	0.2	EX	MP	SN IIIII	BA III	WB ***	BC M	EX
	0.4	EX IIII	MP	SN IIIII	BA	WB **	BC ==	EX
distance	0.6	EX	MP	5N	BA	WB	BC M	EX
Eg.	0.8	EX	MP	SN	BA	WB **	BC MP	EX
dis	1.0	EX SIG	MP 500	SN STE	BA STE	WB	BC SE	EX
	0.1				BA	WB *		EX
	0.2	EX	MP	SN	BA BA	W8		EX
	0.4			SN III		WB *		
94	0.6	1000	MP	T. C.				EX
distance	Attacked	EX	MP	SN	BA	WB 1	BC MA	EX
distance	0.8	EX IIIII	MP	SN	BA	WB	BC MAR	EX
	1.0	EX SMI	MP SM	SN SN	BA MI	WB	BC M	EX
	0.1	EX	MP	SN IIII	BA	WB	BC B	EX
	0.2	EX IIII	MP	SN IIII	BA	WB	BC M	EX
e e	0.4	EX	MP	SN IIII	8A	WB	BC ===	EX 📉
distance	0.6	EX	MP	SN IIII	BA	WB	BC ==	EX E
distance	0.8	EX	MP	SN III	BA III	WB	SC MA	EX
	1.0	EX S	MP W	SN SN	BA M	WB	BC M	EX
	0.1	EX IIII	MP	SN IIIII	BA IIII	WB 1	BC -	EX
	0.2	EX	MP	SN	BA	WB	BC ===	EX
et	0.4	EX	MP	SN	BA	WB	BC M	EX
Wavelet	0.6	EX IIII	MP = 0	SN IIII	BA S	WB *	BC MP	EX
3	1.0	EX IIII	MP MAR	SN BEST	BA BES	WB	BC WW	EX
	0.1		MP					EV
	0.2	EX EX	MP	SN SN	BA BA	WB WB	BC ===	EX
	0.4	EX IIII	MP	SN III	BA	WB	BC -	EX
	0.6	EX	MP	SN	BA .	WB.	BC AP	EX
	0.8	EX IIII	MP	SN IIII	BA	WB -	BC	EX
	1.0	EX IIII	MP	SN HIDT	BA III	WB	BC M	EX
	500	EX	MP	SN	BA	WB	BC -	EX
60	100	EX	MP	SN IIII	BA	WB**	BC M	EX
ion	50	EX IIII	MP	SN IIII	BA	WB	BC ===	EX
iterations	10	EX M	MP	SN M	RA MI	WB	BC M	EX
ii.	1	EX ST	MP ST	SN MI	BA SE	WB	BC M	EX
	average	EX	MP	SN	BA	WB==	BC -	EX

Table 9: Retention of Performance Trends with Varying Thresholds for 1toN_32

				MPI_Bcast	8		do_work
	no loss	EX	MP	CM	CO	LB	EX
	0.1	EX	MP	CM	CO	LB	EX
	0.2	EX EX	MP	CM	CO	LB	EX E
٥	0.4	EX	MP	CM	CO	LB	EX E
difference	0.6	EX E	MP	CM	co	LB III	EX E
differen	0.8	EX	MP	CM	CO	LB	EX
-	1.0	EX S	MP	CM	CO	LB	EX
	10	EX	MP	CM	CO	LB	FX
	100	FX T	MP	CM	CO	LB	EX
d)	1000	EV	MP	CM	CO	LB	EX
- Duc	10000	EX 🚟	MPIER	CM:	co:=	LB	EV
difference	100000	EX 200	MP 200	CM	CO	LB	EX
÷	1000000	EX 200	MP 355	CM	CO	LB	EX IIII
124	0.1	EX	IVIP	CIVI	CO	LB	
	0.2	60	MP	CM	50	LB	EX
:	0.4	EX		CM	CO		EX
a.	0.6	EA	MP	LM	CO		EX.
distance		EX	MP	CM	CO	LB	EX
distance	0.8	EX	MP	C.M	CO	LB	EX
8	1.0	EX	MP See	CM	CO	LB	EX
	0.1	EX	MP	CM	CO	LB	EX
	0.2	EX	MP	CM	CO	LB	EX
0.5	0.4	EX	MP	CM	CO	LB	EX
distance	0.6	EX EX	MP	CM	CO	LB	EX
distance	0.8	EX	MP	CM	CO	LB	EX
100	1.0	EX BO	MP	CM	CO	LB	EX
	0.1	EX	MP	CM	CO	LB	EX
	0.2	EX	MP	CM	CO	LB	EX E
885	0.4	EX	MP	CM	CO	LB	EX E
distance	0.6	EX	MP	CM	CO	LB	EX E
distance	0.8	EX	MP	CM	CO	LB	EX E
D	1.0	EX 255	MP 355	CM	CO	LB	EX
	0.1	EX	MP	CM	CO	LB	EX
	0.2	EX	MP	CM	CO	LB	EX E
-	0.4	EX	MP	CM	CO	LB	EX
Wavelet	0.6	EX MIN	MP	CM	CO	LB	EX E
Va	0.8	EX.	MP	CM	CO	LB	EX
	1.0	EX	MP	CM	CO	LB	EX
	0.1	EX	MP	CM	CO	LB	EX
	0.2	EX	MP	CM	CO	LB	EX E
	0.4	EX.	MP	CM	CO	LB	EX
	0.6	EX	MP	CM	CO	LB	EX
	0.8	EX	MP	CM	CO	LB	EX
	1.0	I X	MP	C.M.	CO	LB	EX
	500	EX	MP	CM	CO	LB	EX
IIS	100	EX	MP	CM	CO	LB	EX
tio	50	EX	MP	CM	CO	LB	EX
iterations	10	EX	MP	CM	CO	LB	EX 🔤
-	1	EX S	MP	CM	CO	LB	EX IIII

Table 10: Retention of Performance Trends with Varying Thresholds for 1to1r_32

		MPI_Ssend					MPI_Recv					do work
	no loss	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS IIII	EX
	0.1	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
	0.2	EX	MP	CM	P2	LR	EX	MP	CM	P2	IS	EX
e,	0.4	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
Relative Difference	0.6	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
Relative	0.8	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
B .O	1.0	EX 1000	MPEN	CMISS	P2 (IIII)	I R	FX III	MP	CMUM	P2 [18]	IS	EX
	10	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
	100	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
g.	1000	EX IIII	MPINE	CMM	P2 1111	LR	EX MI	MP	CMINI	P2 101	LS	EX T
Absolute Difference	10000	EX III	MP	CM	P2 1111	I.R	EX III	MP	CMINE	P2 1111	LS .	EX T
ffer	100000	EX III	MPINE	CMIN	P2 1000	LR	EX CIT	MP	CMDIN	P2 1111	15	EX
A G	1000000	EX IIII	MPIGE	CMBEE	P2 600	I R	FX III	MP	CMON	P2 1	IS	EX III
	0.1	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
	0.2	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
=	0.4	EX	MP	CM	P2	LR I	EX *	MP .	CM.	p2	IS	FX T
Manhattan distance	0.6	EX DI	MPINE	CMINE	P2 1111	I R	EX III	MPET	CMINI	P2 153	IS	EX
tan ta	0.8	EX DE	MPINE	CMTH	P2 1411	LR	EV I'I	MPINE	CAUSES	D2 353	LS	EX
E S	1.0	EX III	MP.M	CMINI	P2 1111	LR	EX X	MP	CNAME	DO WITH	LS	EX
	0.1	EX	MP	CM	P2	LR	FX	MP	CM	P2	IS	EX
	0.2	EX	MP	CM	P2	LR	FX	MP	CM	P2	IS	EX
	0.4	EX DI	MP	CMINI	P2 1111	LR	EX CI	MP	CMMIN	P2 12-1	IS	EX
ean	0.6	EX III	MPINI	CMM	P2 141	LR	EX IO	MP	C. IVENUES	P2 100	IS	EX
Euclidean	0.8	EX DE	MPINE	CMM	P2 1411	LR	EX III	MP	CALIFORNIA	P2 1011	IS	FX
Eu dis	1.0	EX DI	MP III	CMINI	P2 341	LR	EX MAIN	MPTO	CNIMAN	P2 101	LS	EX
_	0.1	EX .	MP	CNA	P2 - MI	I R	EX	MP	CM	P2	LS	EX
	0.2	EX	MP	CM	P2	LR I	EX *	MP •	CM	P2 •	IS	EX
>	0.4	EX 31	MPINE	CNATAR	D2 1411	LR	EX X	MP	CM	P2 955	LC	EV
Chebyshev distance	0.6	EX DE	MPSH	CNATURE	no tata	LR	EX NUM	A A CO DESCRIPTION	CARINE	P2 9500	LS	EX
Chebyshe distance	0.8	EX III	MPDEE	CAATMET	P7 1011	LR I R	CV SEE	MP	C AA MANUE	P2 N/11	LS	EX
S S	1.0	EX BET	MPER	C. IVI PARTIE	DO MAN	LR	EX III	MP	CAUDIN	DO THE	LS	EX
_	0.1	EX	-	CM	P2	LR	FX		CM	P2	IS	EX
	0.2	EX	MP	CM	P2 P2	LR	EX .	MP ·	CM *	P2 •	LS	EX
00000000	0.4	EX 141	MPINE	CM:II	P7 [11]	LR	EX I	MP	CMICI	P2 1	15	EX
Average	0.6	EX III	MPINE	CMM	P2 [H]	I R	EX E	MP	CMILL	P2 101	IS	EX
lavi	0.8	EX III	MPINE	CMI*I*	P2 141	LR	EX III	MP	CMMI	P2 101	LS	EX
4 5	1.0	EX III	MPINI	CMMI	P2 141	LR	EX I	MP	CMMI	P2 101	LS	EX
	0.1	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
elet	0.2	EX	MP	CM	P2	LR	EX .	MP .	CM *	P2 •	LS	EX
ave	0.4	EX M		CMITT	P2 1111	LR	EX III	MP	CMINI	P2 1011		EX
Haar Wavelet	0.6	EX III		CMITTI	P2 [H]	LR	EX III	MP	CMMI		LS	EX
Наа	0.8	EX III	MP	CMIH	P2 141	LR	EX 1	MP #	CM	P2 ##1		EX
	1.0	EX III	_	CMM	-	LR	EX C1	MP	CMMIN	P2 12 1	_	EX
	500	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
us.	100	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
p k	50	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
Keep k iterations	10	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
	1	EX CIT	MPER	CMIE	P2 EEE	LR	EX I	MP	CM	P2 1415	LS	EX
	average	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX

Table 11: Retention of Performance Trends with Varying Thresholds for 1to1s_32

		MPI_Ssend				MPI_Recv					do work	
	no loss	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
	0.1	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
	0.2	EX	MP	CM	P2	LR	EX	MP	CM	10 No. 10	LS	EX
a.	0.4	EX	MP	CM	P2	LR	EX	MP	СМ	P2	LS	EX
Relative Difference	0.6	EX	MP	CM	P2	LR	EX	MP	СМ		IS	EX
Relative Differen	0.8	EX	MP	CM	P2	LR	EX	MP	CM		IS	EX
A .O	1.0	EX III	MPINE	CMINI	P2 THE	I R	FX IIII	MPILE	CMINA	P2 IIIII	S	FX
	10	EX	MP	CM	P2	I R	EX	MP	CM	P2	S	FX
	100	FX III	MP	CM	P2	I R	FX	MP	CM	P2	S	FX T
- 64	1000	EX ISSE	MPTOON	CMIM	P2 1018	LR	EX DE	MP IN	CMIN	P2 HG	IS	FX
Absolute Difference	10000	EX MI	AIP WIN	CM	97 MIN	LR	FX III	MP	EMILL	P2	S	FX
fer	100000	EX III	MPINI	CMITT	PZ INI	LR	FY III	MPIII	CMIIII	P2 IIII	15	EX
A O	1000000	EX MA	MP	CMIE	PZ INS	LR	EV III	MP	CMSSS	po litta	LS .	EX
	0.1	EX	MP	СМ	P2	LR	EX	MP	CM	-	LS	EX
	0.2	EX	MP	CM	P2	LR	EX	MP	CM		LS	EX
E	0.4	EX	MP	CM	P2	LR	EX	MP	CM	C. C. C. S.	LS	FX
Manhattan distance	0.6	EX ION	MP	CM	P2 1558	LR IR	EX III	MP IN	CMIN	100	15	EX
tan tan	0.8	EX IOIS	MP DE	CM	DY MAKE	LR	EX DE	MP IN	CMIN		LS	EX
Ma	1.0	EX IDIA	MP ISTE	CMIDIE	P2 1005	LR	EX DE	MP III	CM In:		LS	EX
_	0.1	EX			P2	LR	EX		CM		I S	EX
	0.2	EX	MP	CM	PZ III	LR	FX	MP	Color demonstration	P2	15	EX
	0.4	EX IDE	MP	CM	P2 1015	LR	EX DE	MP III	CMIT		IS	EX
Euclidean	0.6	EX ION	MP DIS	CMIDIO	PZ INTE		EX DE	MP IN	CMINE		S	
Euclidear distance	0.8	EX ION	MP DE	CMINNE	PZ IDIB	LR	EX DE	MP INC	CMINE	P2 8888	IS	EX EX
Euc	1.0	EX 1016	MP IOIS	CALIBRA	PZ IMIN	LR IR	EX In	MP IU	CMINE	P2 100	LS	EX
-	0.1		MP WW	CM	PZ IIIII	LR I R	EX		CM	P2 P4	LS	
	0.2	EX	MP dille	CM	P2	LIX.	-	MP	53.00		-	EX
>		EX III	MP III	CMISSE	P2 100	LR	EX In:	MP	CM		LS	EX
Chebyshev distance	0.4	and the second	MP ISS	CM	P2 111	LK	EX Inc	MP III	CHARMS	PZ BAIR	LS	EX
Chebyshe distance	0.8	EX IONS		CM	PZ III	LR	EX DE	1011	CMINE	P2 8888	LS	EX
dist	1.0		MPISS			LR	EX IND	MPIN	CMIT	P2 1819	LS	EX
	0.1	EX E	MP	CM	PZ INT	LR	Name of Street	1411	C. P. STREET		LS	EX
	0.1	EX	MP	CM	P2	LR	EX	MP	CM		LS	EX
-00-11-0-1	0.4	EX CO	MP CC	CM	P2 ISSE	LR	EX Iui	MP III	CMIN	-	LS	EX
er er	0.6	EX 1018	MPICIU	CM	P2 1055	I R	EX III	MP III	CMINE		15	EX
Average	0.8	EX DID	MP	CM	P2 1005	LR	EX In	MP III	CMIN		IS	EX
₹ ≥	1.0	EX I	MP DE	CMISS	P2 1005	LR	EX In	MP III	CM In:		IS	FX
	0.1	EX	MP	CM	P2	LR	EX	MP	CM	1 2 2	LS	EX
let	0.2	EX	MP	CM	P2	LR	EX	MP	CM		LS	EX
ave	0.4	EX 1018		CM	P2 1015	LR	EX In:		CMTH		LS	EX
×	0.6	EX III	MPINE	CM	P2 1000	LR	EX In:	MP IU:	CM In:		LS	EX
Haar Wavelet	0.8	EX IDIO	MP III	CMH	P2 1005	LR	EX IN		CM III	P2 BB	LS	EX
<u></u>	1.0	EX III	MP III	CM	P2 1000	LR	EX DE	MPIN	CMIN	P2 IIIi	LS	EX
	500	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
(4)	100	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
k tion	50	EX	MP	CM	P2	LR	EX *	MP *	CM*	P2 *	LS *	EX E
Keep k iterations	10	EX	MP	CM	P2	LR	EX	MP	CM		LS	EX 🔤
≥ .=	1	EX IN	MPINE	CMINI	P2 IN	LR	EX I	MP	см	PZ ME	LS	EX
	average	EX	MP	СМ	P2	LR	EX	MP	CM	P2	LS	EX

 $Table~12: Retention~of~Performance~Trends~with~Varying~Thresholds~for~Nto1_1024$

				MPI_Gathe	r		do_work
	no loss	EX	MP	CM	CO	ER	EX IIII
	0.1	EX III	MP	CM	CO	ER	EX
	0.2	EX I	MP	CM	CO	ER	EX
ø.	0.4	EX.	MP	CM	co	ER	EX IIII
difference	0.6	EX MINI	MP	CM	CO	ER .	EX
differen	0.8	EX III	MP	CM	co	ER	EX
0	1.0	EX MIN	MP	CM	CO	ER III	EX
	10	EX	MP	CM	CO	ER	EX IIII
e e	100	EX	MP	CM	CO	ER	EX
	1000	EX .	MP	CM	co	ER	EX
eno eno	10000	EX	MP	CM	CO	ER	EX
absolute difference	100000	EX MI	MP MAN	CM	CO	ER	EX
-E	1000000	EX Ma	MP MAN	CM	CO	ER	EX
	0.1	EX	MP	CM	co	ER	EX
	0.2	EX	MP	CM	co	ER	EX
=	0.4	EX	MP	CM	co	ER	EX
e	0.6	EX	MP	CM	co	ER ER	EX
Manhattan distance	0.8		MP	CM	CO	ER	EX
	1.0	EX			CO		
	0.1	EX MA	MP	CM		ER	EX
		EX	MP	CM	CO	ER	EX
	0.2	EX	MP	CM	CO	ER	EX
Euclidean	0.4	EX	MP	CM	co	ER	EX IIII
anc	0.6	EX	MP MP	CM	CO	ER	EX
distance	0.8	EX	MP	CM	CO	ER	EX IIIII
	1.0	EX M	MP	CM	co	ER	EX
	0.1	EX	MP	CM	CO	ER	EX
	0.2	EX	MP	CM	CO	ER .	EX
distance	0.4	EX M	MP	CM	CO	ER	EX
distance	0.6	EX	MP	CM	CO	ER	EX IIII
ste	0.8	EX M	MP	CM	CO	ER	EX
	1.0	EX M	MP MAIN	CM	CO	ER	EX
	0.1	EX E	MP	CM	CO	ER	EX
	0.2	EX	MP	CM	CO	ER	EX
t to	0.4	EX	MP	CM	CO	ER	EX
Wavelet	0.6	EX	MP	CM	CO	ER	EX
Wavelet	0.8	EX	MP	CM	CO	ER	EX
	1.0	EX	MP	CM	CO	ER	EX
	0.1	EX	MP	CM	co	ER	EX
	0.2	EX	MP	CM	CO	ER	EX
	0.4	EX EX	MP	CM	CO	ER ER	EX EX
	0.8	Sec. C. S.	MP	CM	CO		EX
	1.0	EX EX	MP	CM	CO	ER ER	EX
	500	EX	MP	CM	co	ER	EX
1001	100	EX	MP	CM	co	ER ER	EX EX
ons	50	EX	MP	CM	CO		EX
ati	10					ER	
iterations	1	EX EX	MP	No. 17.1	CO	ER	EX
CANCEL SECTION	average	EX	MP MP	CM	CO	ER ER	EX

Table 13: Retention of Performance Trends with Varying Thresholds for NtoN_1024

	1			MPI	Barrier			do_work
	no loss	EX	MP	SN =	BA CO	W8	BC -	EX
	0.1	EX	MP	SN	BA I	WB	BC	EX
	0.2	EX	MP	SN =	8A ===	WB	BC P	EX I
e.	0.4	EX	MP	SN =	8A	WB	BC B	EX I
difference	0.6	EX E	MP	SN	BA	WB ==	BC B	EX E
fer	0.8	EX	MP	SN I	BA B	WB.	RC III	EX
differen	1.0	EX M	MP	SN M	BA MA	WB	BC MI	EX
	10	EX	MP	SAL	BA B	WB =	BC	FX
	100	EX C	MP	SN	RA T	WB =	HC B	FX
a)	1000	FX	MP	SN I	RA SE	WB =	RC III	EX E
- Suc	10000	EX	MP	SM	RA B	WB III=	BC BE	EX
difference	100000	EX M	MP	SN III	BA BA	WB	BC M	EX
. <u>E</u>	1000000	EX EX	MP	SN =	BA BA	WB	BC M	EX
	0.1	EV	MP	CNI	RA .	WB =	00	EV
	0.2	EX.	MP	SN	RA I	WB ==	200	EV
	0.4	EV.	MP MP	SAL	RA S	WB =	DC I	EV
distance	0.6	EV	MD	CM	BA BA	WB =	Dr.	EV
5	0.8	EX	10112	SN SN	BA		214	EX
dis	1.0	-	MP	-	-	WB =	00	- Back
	0.1	EX MA	MP	SN MA	BA BI	WB	BC M	EX
	-	EX	MP	SN	BA B	WB =	BC -	EX
	0.2	EX	MP	SN	BA =	WB =	BC	EX
e e	0.4	EX	MP	SN =	BA 📉	WB =	HC -	EX
distance	0.6	EX 🔤	MP	SN M	BA I	WB =	BC	EX
distance	0.8	EX	MP	SN =	BA ===	WB =	BC	EX E
	1.0	EX M	MP	SN m	BA in	WB	BC I	EX
	0.1	EX ME	MP	SN MT	BA M	WB	BC	EX
	0.2	EX ==	MP	SN ===	BA III	WB =	EC.	EX
e e	0.4	EX =	MP	SN ===	BA ===	WB =	BC -	EX I
Suc	0.6	EX	MP	SN man	BA	WB ==	AC .	EX
distance	0.8	EX M	MP	SN ME	BA ME	WB =	EC .	EX
	1.0	EX	MP	SN SN	BA MA	WB	BC M	EX
	0.1	EX	MP	SN SN	BA	WB =	BC E	EX
	0.2	EX	MP	SN I	BA I	WB =	BC	EX
et a	0.4	EX.	MP	SN	BA	WB ==	BC E	EX
Wavelet	0.6	EX	MP	SN	BA BA	WB _	BC	EX
W	1.0	EX	MP	SN	BA	WB ==	NC -	EX
	0.1	EX	MP	SN	BA.	_	ac I	EX
1	0.2	EX	MP	SN	BA BA	WB =	BC	EX
	0.4	EV	MP	SN	BA	WB =	BC F	EX
	0.6	EX	MP	SM	BA	WB =	RC BE	EX E
	0.8	EX	MP	SM	BA	WB ===	BC B	EX
	1.0	£X/	MP	SN	BA BE	WB	BC C	EX E
	500	EX	MP	SN	BA	WB ==	BC =	EX
60	100	EX WIIII	MP	SN IIII	BA ·	WB =	BC B	EX
ion	50	EX IIII	MP	SN IIII	BA	ws E	BC	EX
i i	10	EX IIIII	MP	SN IIII	BA	WB _	HC F	EX
iterations	1	EX M	MP	SN 🔤	BA ME	WB	BC ME	EX
	average	EX	MP	SN	BA	WB	BC ==	EX

Table 14: Retention of Performance Trends with Varying Thresholds for 1toN_1024

				MPI_Bcast	si		do_work
	no loss	EX	MP	CM	CO	LB	EX.
	0.1	EX	MP	CM	CO	LB	EX
	0.2	EX EX	MP	CM	CO	LB	EX E
94	0.4	EX EX	MP	CM	co	LB	EX E
relative	0.6	EX E	MP	CM	co	LB	5 X .
relative differen	0.8	EX	MP	CM	CO	LB	EX E
5.2	1.0	EX M	MP	CM	CO	LB	EX
	10	EX	MP	CM	co	LB	EX
	100	EX	MP	CM	co	LB	EX.
	1000	EX E	MP	CM	co	LB	EX E
ute	10000	EX MI	MP	CM	co	LB	EX E
absolute	100000	EX M	MP ***	CM	CO	LB	EX
B :5	1000000	EX	MP	CM	CO	LB	EX
	0.1	EX	MP	CM	CO	LB	EX
	0.2	EX.	MP	C'hA	CO	LB	FX
E	0.4	FX	MP	CM	CO	LB	EX E
atte	0.6	EX	MP	CM	CO	LB	EV E
Manhattan distance	0.8	EV B	AAD	CM	COMM	LB	EX
	1.0	EX	MP	CM	co	LB	EX
	0.1	EV	MP	CM	CO	LB	EX
	0.2	EV	MP	CM	co	LB	EV E
20	0.4	EV	MP	CM	CO	LB	E-0
ean	0.6	EX	MP	CM	CO	LB	
Euclidean distance	0.8	EX	NAD WAR	CM	CO	LB III	EA
dis di	1.0	EX ***	14P	CM	CO	LB	EV
	0.1	EX	MP	CM	CO	LB	EX
	0.2	EA	DATE OF THE PARTY	CM	CO	LB III	EX
>	0.4	EA .	AST .	CAL	co	LB III	EX
Chebyshev distance	0.6	10	MP -	CM	CO	LB III	EX
Chebysh	0.8	EA.	4011	CM	CO	LB III	EX
dist	1.0	EV.	MP	CM	CO		EX
	0.1	EX	MP	CM	CO	LB	EX
	0.2	EX	MP	CM	CO	LB	LX:
	0.4	EA	MP	CM	CO	LB	EX
Average	0.6	EX	MP	CM	CO	LB	EX
ave	0.8	EX W	MP	CM	COM	LB	FX E
₹ ≥	1.0	EX MIN	MP	CM	CO	LB	EX.
	0.1	EX	MP	CM	CO	LB	EX E
let	0.2	EX	MP	CM	CO	LB	EX E
ave	0.4	EX	MP	CM	CO	LB	EX.
Haar Wavelet	0.6	EX E	MP	CM	CO	LB	EX
laa	0.8	EX EX	MP	CM	CO	LB	EX
-	1.0	EX EX	MP	CM	co	LB	EX
	500	EX	MP	CM	CO	LB III	EX
35	100	EX	MP	CM	CO	LB	EX
X tio	50	EX	MP	CM	CO	LB III	EX E
Keep k iterations	10	€X	MP	CM	CO	LB III	EX
× .=	1	EX I	MP	CM	CO	LB	EX
	average	EX	MP	CM	CO	LB TT	EX

Table 15: Retention of Performance Trends with Varying Thresholds for 1to1r_1024

		MPI_Ssend					do work					
	no loss	EX 1	MP*	CM*	P2 *	LR *	EX	MP	СМ	P2	LS	EX
	0.1	EX *	MP*	CM"	P2 1	LR	EX	MP	CM	P2	LS	EX E
	0.2	EX *	MP1	CM"	P2 1	LR	EX	MP	CM	P2	LS	EX E
9	0.4	EX 1	MP"	CM*	P2 *	LR .	EX	MP	CM	P2	LS	EX III
ive	0.6	EX T	MP"	CM*	P2 1	LR	EX	MP	CM	P2	LS	EX
Relative Difference	0.8	EX 1	MP1	CM*	P2 *	LR	EX	MP	CM	P2	LS	EX
S io	1.0	EX III	MPIN	CMINE	P2 1010	LR	EX THE	MPERE	CMFIRE	P> (200)	LS	EX
	10	EX *	MP1	CM*	P2 *	LR	EX	MP	CM	P2	LS	EX III
	100	EX ***	MP	CMITTER	P2 ***	LR "	EX	MP	CM	P2	LS	EX 🔤
	1000	EX Inte	MP I	CMINI	P2 1010	LR	EX KIE	MPIE	CMME	P2 1016	LS .	EX EX
en	10000	EX MIN	MPIN	CMINE	PZ MIN	LR	EX THE	MP III	CM ₩₩	P2 100	LS .	EX .
Absolute Difference	100000	EX III	MPHH	CMUR	P2 (III	LR	EX UNIT	MPUNE	CMUNIC	P2	LS	EX .
A G	1000000	EX EE	MP	CME	P2 (88)	LR	FX BES	MPERE	CMEN	P2 商店	LS	EX
	0.1	EX *	MP!	CM*	P2 1	LR	EX *	MP	смаш	P2 *	LS	EX
	0.2	EX 1015	MP	CMINE	P2 5555	LR	FX	MP	CM	P2	LS	EX
an	0.4	FX ***	MP	CM	P7 ****	LR	FX	MP	CM	P2	15	FX
Manhattan distance	0.6	EX 1113	MPILE	CMMIN	P2 188	I R	EX SI	MPER	CMISH	P2 115	15	EX
	0.8	EX III	MPHE	CMILE	P2 1885	I R	EX III	MPINE	CMIN	P2 1011	15	EX
≥ 5	1.0	EX INS	MP.	CMILE	P2 1815	I R	EX III	MPINE	CMMI	P2 1116	15	FX
	0.1	EX 1	MP*	CM1	P2 1	LR	EX 1	MP	CM	P2 1	IS	FX S
	0.2	EX MIN	MP1015	CMINE	P2 1515	LR	EX .	MP	CM	P7	IS	EX
_	0.4	EX II :	MPME	CMINE	P2 1118	LR	EX TOP	MPER	CMITE	P2 106	15	FX
Euclidean distance	0.6	EX 1	MPILE	CMILE	P2 1188	LR	EX III	MPTH	CMINE	P2 108	15	EX
Euclidear distance	0.8	EX 1000	MPINE	CMINI	P2 1415	I R	EX III	MPTH	CMILL	P2 1011	15	FX S
B. E	1.0	EX 1000	MPINI	CNATHON	P2 105	I R	EX IOI	MPTH	CMMI	P2 10B	LS III	FX
	0.1	EX I	MP	CNASSIN	DO ISSUE	I R	EV 1	MP	CM	P2 1	I C	FX
	0.2	EX I	MP *	CM	D2 1	I R	EX *	MP *	CM	P2 *	15	EX
>	0.4	EX ME	MPDE	CMINE	DO THE	LR	EX UII	MP IOI	CMUE	p2 PH	15	EV
Chebyshev distance	0.6	FX MIS	MPINE	CMINI	po fata	LR	EX III	MP	CMILE	po 111;	I C	EV
tan	0.8	EX MIN	MPMS	CASTIN	P2 1158	LR	EX III	MP III	CMMB	D2 1111	15	EV
5 %	1.0	EX ITE	MP	CM	P2 (1991)	LR	EX DES	MP DE	CMEN	P2 500	IS	EV
	0.1	EX *	MP*	CM	po film	LR	EX.	MP	CNA	p2 1	IS	EX
	0.2	EX ** *	MP	CM	P2 ***	LR	FY	MP	CM	P2	LS	EX
39/12/2	0.4	EX 11 :	MPHI	СМНИ	P2 368	LR	EX ICE	MPTOB	CMICE	P2 1116	LS	EX
Average	0.6	EX MIN	MPIN	CMMM	P2 1815	LR	EX THE	MPINE	CM	P2 1016	LS	EX
Vav	0.8	EX IIII	MP	CMIMI	P2 1100	LR	EX III	MP	CMMI	P2 1015	LS	EX
2 >	1.0	EX III	MP	CMP	P2 1011	LR	EX EE	MPRO	CMUL	P2 10E	LS .	EX
40	0.1	EX ***	MP	CM	P2 ****	LR '	EX	MP	CM	P2	LS	EX
Haar Wavelet	0.2	EX **	MP	CM	P2 ***	LR	EX	MP	CM	P2	LS	EX
Vav	0.4	EX Mile	MPILU	CMMIN	P2 14 15	LR	EX III	MPINE	CMILL	P2 1016	LS	EX
N JE	0.6	EX III	MP	CM	P2 166	LR	EX III	MPER	CM	P2 10B	LS	EX
На	0.8	EX III	MP	CM	P2 1115	LR	EX III	MPILE	CM	P2 10B	LS	EX
	1.0	EX MI	MP 1011	CM	P2 1111	LR	EX ICE	MPROIS			LS	EX
	500	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
IIIS	100	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
p k atio	50	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
Keep k iterations	10	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
-	1	EX RE	MP	€M###	P2 ###	LR	EX III	MP	CMPE	P2 (15)	LS	EX
	average	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX

Table 16: Retention of Performance Trends with Varying Thresholds for 1to1s_1024

		MPI_Ssend					MPI_Recv					
	no loss	EX *	MP*	CM*	P2 *	LR	EX *	MP	CM	P2 ****	LS	do_work
	0.1	EX *	MP1	CM"	P2 1	LR	EX *	MP	CM*	P2	LS	EX
	0.2	EX *	MP1	CM"	P2 1	LR	EX *	MP*	CM	P2 ***	LS	EX
a.	0.4	EX 1	MP*	CM*	P2 *	LR .	EX *	MP 1	CM*	P2 *	LS	EX
Relative Difference	0.6	EX 1	MP*	CM*	P2 1	LR	EX *	MP	CM*	P2 *	LS	EX
Relative	0.8	EX *	MP1	CM*	P2 *	LR	EX 1	MP *	CM*	P2 *	LS	EX
A O	1.0	EX III	MP III	CMINI	P2 1988	LR	FX BRE	MPER	CMITTE	P2 SHE	LS	EX
	10	EX *	MP*	CM*	P2 1	LR	EX .	MP*	CM*	P2 ***	LS	EX
	100	EX ***	MP	CMIT	P2 "	LR	EX 1	MP*	CM	P2 ***	LS	EX
	1000	EX PIL	MP III.	CM ² T	P2 Im.	LR *	EX HIL	MP MM	CMBM	P2 114	LS	EX
ute	10000	EX PIL	MPER	CMPT	P2 191	LR *	EX IIII	MP SEE	CMISS	P2 1881	LS	EX
Absolute Difference	100000	EX FILE	MPIGH	CMIN	P2 1018	I R	FX ME	MPRE	CMINE	P2 III	15	FX
A id	1000000	FX MIN	MPSEE	CMILL	P2 暗珠	I R	FX REL	MPSE	CMEN	P2 REE	IS	FX
	0.1	EX 1	MP*	CM*	P2 1	LR	FX *	MP	CMI	P2 ***	LS	EX
	0.2	EX MILL	MP May	CMINE	P2 5500	LR	FY 5	MP	CMI	P7 N	IS	EX
E .	0.4	FX 11	MP	CMITTER	P7 M	LR	EX 1	MP*	CM	P2 ****	LS	FX
atte	0.6	EX PIL	MPER	CMINE	P2 191	LR	EX HIT	MP SET	CM	P2 1111	IS	EX
Manhattan distance	0.8	EX TIL	MPTER	CMINE	P2 190	LR *	EX SI	MP	CM	P2 1111	LS	EX
A Si	1.0	EX III	MP	CMIN	P2 11.	LR *	EX IIII	MP	CM	P2 1111	LS	EX
_	0.1	EX I	MP	CM	DO SININ	LR	EX I	MP	CM	P2 1	IS	EX
	0.2	EX **	MP	CIVI	P2 MIII	LR	EX *	MP	CM*	P2 ****	IS	EX
	0.4	EX PI	MPROU	CMIT.	P2 11.	LR ·	EX HIL	MPSH	CM SIT	P2 119	LS	EX
Euclidean	0.6	EX HILL		CMITT	P2 1111	S.I.	EX HILL	MP 575	CMBIN	P2 1111	15	
Euclidear	0.8	EX HIS		CM	P2 100		EX IIII	MP 5	CMIN	P2 1111	IS IS	EX
dis dis	1.0		MPER	CM	P2 IIII	bet 5	EX SIL		CMBB	P2 1111	IS I	EX
-	0.1	EX III		CMMIN	PZ MIN	F17		MPUH			IS	EX
	0.2	LA Britain	1411	CM	PZ IIIII	LR	EX	MP	CM	P2	No.	EX
		340	MP	CM	PZ IIII	LR	EX	MP	CM	P2 *	LS	EX
Chebyshev distance	0.4	EX ***	MP	CM-11	PZ KIN	LR	EX HU	MP SE	CM	P2 55	LS	EX
Chebyshe distance	0.6	EX PL	MPCL	CMMM	PZ KING	LR			CM	-	LS	EX
Che	1.0	EX 15E	MP	CMARKE	PZ MBS	LR	EX BU	MP SIM	CMSS	P2 99	LS	EX
		EX MIN	MP	CM	PZ BIN	LR	EX ME	MP	CMRE	PZ ME	LS	EX
	0.1	EX .	MP*	CM	PZ **	LR	EX	MP	CM	P2 ****	LS	EX
-00/11/2/7	0.4	EX MI	MP III	CMIN	P2 31	LR	EX SET	MP SOT	CM	P2 1111	LS	EX
let	0.4	EX TU	MP	CMINE	PZ EII	LR *	EX SIS	MP III	CM	P2 1111	LS LS	FX
Average	0.8	EX TI	MP	CM	P2 191	LR *	EX HILL	MP 1111	CMN	P2 55	LS	EX
₹ 3	1.0	EX III.	MP	CM II	P2 11	LR *	EX SU	MP 444	CM !!!!	P2 1111	LS	EX
	0.1	EX M	MPMW	CMIN		LR	EX 1	MP.	CM	P2 *	LS	EX
let	0.2	EX 1	MP	CM*	P2 1	LR	EX *	MP 1	CM.	P2 *	LS	EX
Haar Wavelet	0.4	EX HI.	MP III	CMINI	P2 34.	LR	EX SON	MP STA	CMBIN	P2 1111	LS	EX
3	0.6	EX III	MP	CM***		LR *	EX SE	MP M	CM	P2 BH	LS	EX
99	0.8	EX III.	MP III	CMIT.	P2 11.	LR *	EX HILL	MP M	CM	P2 \$50	LS	EX
	1.0	EX III.	MPHIL	CMIT	P2 111	_	EX III	MP	CM	P2 1111	LS	EX EX
	500	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
8	100	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX 🔤
k io	50	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX
Keep k iterations	10	EX	MP	CM	P2	LR	EX	MP	CM	P2	LS	EX 📉
× .=	1	EX P	MP	CM	P2 (20)	LR	EX WE	MPER	CM	PZ ME	LS .	EX
	average	EX	MP	CM	P2	LR	EX	MP	СМ	P2	LS	EX

Table 17: Retention of Performance Trends with Varying Thresholds for sweep3d_8p

	T .			pmpi	_recv_			sweep_
	no loss	EX	MP	CM	P2	LS	MO E	EX CONTRACTOR
	0.1	EX	MP	CM	P2	L5 Days	MO B	EX DESIGNATION
	0.2	EX	MP	CM	P2	LS B	MO .	EX CONTRACTOR
e e	0.4	EX-	MP	CM	92	LS CONTRACTOR	MOUTH	EX EX
relative	0.6	EX	· MP	CV	P2	LS	MO	EX TOTAL
difference	0.8	EX	MP	CM	P2	LS B	MO BE	EX HEREN X3
ē	1.0	EX	MP	CM	P.2	L5	MC	EX
	10	EX	MP	EM	P2	LS III	MONINE	EX III
	100	EX	MP	CM	P2	LS	MO	EX CONTRACTOR
9	1000	EX	MP	CM	92	LS	MO:	EX III
difference	10000	EN INC.	- AIP	CV	P2	L5 10 10 10 10 10 10 10 10 10 10 10 10 10	MCM1 - BIS	AX DIS BOOK
difference	100000	EX	MP	CM	P2	L5 []	MOE CONTRACTOR	EX
Ē	1000000	EX	MP	CM	P2	LS	MO	EX
7 121	0.1	EX:	MP	CM	P2	LS	MO	EX
	0.2	EX	MP	CW	P2 -	LS	MOST	EX DESCRIPTION
	0.4	EX	MP	. CM	P2	LS	MO	EX
distance	0.6	EX	MP	CM	99	LS	MO B	EX
Eg.	0.8	EX	MP	CM	P2 -	LS	MODIFICATION	EX
Manhattan distance	1.0	EX	MP	CM	P2	L5	MO B	EX
	0.1	EX	MP	CM	99	LS	MO	EX
	0.2	EX	MP	EM	P2.	15	MO	EX
	0.4	EX	MP	CM	02	IS TO THE RESERVE OF THE PERSON OF THE PERSO	MC	EX
8	0.6	EX	640	CM	00	16	MO	EX
distance	0.8	EX	600	E3/	00	LS	MO	EX
distance	1.0	EX	MI MI	CM	03	LS	MO	EX
	0.1	EX	MD	CV	00	LS	MO	EX
	0.1		No.	CM	00			EX
		EX	D.F.	CM	62	LS	MO	EN
e e	0.4	EX	MP	CM	000	T2	MO	CA
distance	0.6	EX	NIP	X, W	P2	LS	MO B	EX
distance	0.8	ex .	MP	CM	92	LS TO THE TOTAL TO	MO II II	EX
	1.0	EX	MP.	CM	P2	L5 E	MUE	EX
	0.1	EX	MP	CM	P2	LS	MO	EX CHARGE
	0.4	EX EX	MP	CM	P2	LS	MO B B	EX
Wavelet	0.6	EX	MP	CM	P2	LS	MO B B	EX
Wavelet	0.8	EX	MP	CM	P2	LS	MO B	EX
3	1.0	EX	MEP	CM	P2	LS	MO	EX
	0.1	EX	MP	CM	P2	LS	MO B	EX
	0.2	EX EX	MP	CM	P2	LS THE REST	MO B	EX .
	0.4	EX	MP	CM	P2	LS	MO .	EX
	0.6	EX E	MP	СМ	P2	LS	MO:	EX
	0.8	EX	MP	CM	P2	LS	MO B	EX CONTRACTOR
	1.0	EX.	MP	CM	P2	L5	MOUNT	EX
	500	EX	MP	CM	P2	1.5	MO B	EX
100	100	EX .	MP	CM	P2	15	MG	EX EX
tion	50	EX	MP	CM	P2	US	MC	EX MINI
era r	10	EX	MP	CM	P2	L5 (1000)	MODE	EX III
iterations	1	EX	MP	CM	P2	L5	MG	EX
	average	EX	MP	CM	P2	L5	MO	EX

 $Table~18: Retention~of~Performance~Trends~with~Varying~Thresholds~for~sweep3d_32p$

				pmp	i_recv_			sweep_
	no loss	EX mil	MP I	CM I	P2	LS -	MO	EX
	0.1	EX I	MP -	CM	P2 ==	LS	MO	EX
	0.2	EX 📶	MP -	CM	P2 -	LS	MO	EX
9	0.4	EX I	MP	CM -	P2	LS	MO	EX
difference	0.6	EX .	MP	CM mil	P2 -	LS	MO	EX
differen	0.8	EX I	MP	CM -	P2	LS	MO	EX I
	1.0	EX	MP	CM	P2	LS Said	MONE	EX
	10	EX I	MP -	CM	P2	15	MO	EX .
	100	EX I	MP	CM	P2 -	LS	MO	EX
e e	1000	EX -	MP -	CM -	P2 -	LS	MO	EX I
enc	10000	EX P	MP	CME	P2 P	LS	MO	EX
difference	100000	EX	MP	CM	P2	LS Sect	MO	EX
₹ ₩	1000000	EX	MP	CM	P2	LS MILL	MON	EX
	0.1	EX	MP -	CM -	P2 -	LS	MO	EX
	0.2	EX I	MP -	CM	P2 -	LS	MO	EX
	0.4	The P. Lindson	MP	CM	P2 P2	LS	MO	EX
distance	0.6	EX I	MP -	CM -	P2 -	LS	11 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	EX
Ē	0.8			100000000000000000000000000000000000000			MO	-
dis	1.0	EX mall	MP mil	CM mil	P2 ===	LS	MO	EX
		EX 🚟	MP	CM mil	P2 -	LS	MO	EX
Euclidean distance	0.1	EX I	MP	CM	P2 ==	LS	MO	EX
	0.2	EX	MP -	CM .	PZ 🔤	LS	MO	EX
	0.4	EX	MP -	CM -	P2 -	LS	MO	EX
distance	0.6	EX I	MP	CM -	P2 -	LS	MO	EX
ist	0.8	EX modi	MP	CM mil	P2 -	LS	MO	EX
. 0	1.0	EX June	MP	CM I	P2	LS	MO	EX
	0.1	EX E	MP ==	CM	P2 ===	LS	MO	EX
	0.2	EX =	MP =	CM =	P2	LS	MO	EX
	0.4	EX 📥	MP -	CM -	P2 ===	LS	MO	EX E
distance	0.6	EX	MP mil	CM -	PZ 📥	LS	MO	EX E
distance	0.8	EX -	MP M	CM -	P2 -	LS	MO	EX
0	1.0	EX	MP	CM	P2	LS SEES	MONEY	EX
	0.1	EX E	MP	CM -	P2 ==	LS	MO	EX
	0.2	EX -	MP	CM-	P2 -	LS	MO	EX
4	0.4	EX I	MP	CM	P2	LS	MO	EX
Wavelet	0.6	EX M	MP	CM	P2 1	LS	MO	EX E
Wavelet	0.8	EX M	MPM	CM	P2 Miles	LS	MO	EX
	1.0	EX M	MP	CM	P2 200	LS	MO .	EX
	0.1	EX ===	MP -	CM I	P2 =	LS	MO	EX
5	0.2	EX I	MP	CM -	P2 mil	LS	MO	EX
	0.4	EX ·	MP -	CM	P2 -	LS	MO	EX
,	0.6	EX M	MPM	CM	P2 1	LS	MO	EX
	0.8	EX P	MP	CM	P2 Marie	LS	MO	EX
	1.0	EX M	MP ME	CM	P2 1	LS	MO	EX
	500	EX I	MP M	CM	P2	LS made	MO	EX
SIIIS	100	EX	MP	CM	P2	LS Seel	MOSE	EX
atio	50	EX	MP	CM	P2	LS SEE	MOUNT	EX
iterations	10	EX	MP	CM	P2	LS 5961	MOSSES	EX
	1	EX	MP	CM	P2	LS Sign	MONE	EX
	average	EX mil	MP	CM	P2	LS	MO	EX