ARMY RESEARCH LABORATORY

Optimization and Parallelization Efforts in Steady-State Particle Tracing in EnSight

Anders Grimsrud

ARL-CR-298

August 1996

prepared by

Computational Engineering International, Inc.
P.O. Box 14306
Research Triangle Park, NC 27709

under contracts

DAAL01-95-P-2215
and
DAAL01-95-P-2275

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE August 1996	3. REPORT TYPE AND DATES COVERED Final, Sep - Dec 95

4. TITLE AND SUBTITLE Optimization and Parallelization Efforts in Steady-State Particle Tracing in EnSight	5. FUNDING NUMBERS C: DAAL01-95-P-2215 C: DAAL01-95-P-2275
6. AUTHOR(S) Anders Grimsrud	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Computational Engineering International, Inc. P.O. Box 14306 Research Triangle Park, NC 27709	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: AMSRL-SC-CC Aberdeen Proving Ground, MD 21005-5067	10.SPONSORING/MONITORING AGENCY REPORT NUMBER ARL-CR-298

11. SUPPLEMENTARY NOTES

Point of contact for this report is Richard C. Angelini, U.S. Army Research Laboratory, ATTN: AMSRL-SC-CC, Aberdeen Proving Ground, MD 21005-5067.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

EnSight is a commercially available, general-purpose visualization program currently used at a variety of DOD and commercial research facilities. The software was originally designed and implemented for the vector-based Cray computer systems. The advent of reasonably affordable parallel computers configured as symmetric multiprocessors has enabled advanced government and commercial research groups to perform computational analyses on solution sets consisting of several million grid nodes. Unfortunately, the ability to generate these massive data sets has outstripped the ability to interactively visualize them. The first goal of this project was to optimize the EnSight software for efficient sreamline generation in a steady-state vector field. After initialization, the streamline generation is a parallel task as each particle path is independent of any other path.

14. SUBJECT TERMS computer graphics, scientific visualization, computational fluid dynamics, parallel programming, Silicon Graphics, Inc.	15. NUMBER OF PAGES 17
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

INTENTIONALLY LEFT BLANK.

ACKNOWLEDGMENT

Computational Engineering International, Inc. (CEI) wishes to thank the U.S. Army Research Laboratory (ARL) for its support and encouragement on this project.

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

INTENTIONALLY LEFT BLANK.

1. INTRODUCTION

In this report we present a summary of our successful efforts to optimize/parallelize the steady-state particle tracing (streamline) algorithm in EnSight. This project was funded by the U.S. Army Research Laboratory (ARL), located at Aberdeen Proving Ground, MD, and is considered completed as of the date of this report. All modifications as a result of this project have been included in the newest commercial release of EnSight, version 5.5.1.

2. BACKGROUND

A streamline visualizes a steady-state vector field by displaying the path that a massless particle would follow if placed in that field. At each point on the particle trace, the direction of the trace is parallel to the vector field at that point and time.

To create the streamline, the user specifies a starting location, known as an emitter point. Given this emitter point, EnSight performs an integration through the vector field using a fourth-order Runge-Kutta algorithm with varying time step. After initialization (where EnSight creates a table describing which computational elements are neighbors), the streamline generation is, in theory, a parallel task as each particle path is independent of any other path.

If the streamline generation can be performed quickly, the user has the ability to look at the particle paths in "real time," meaning he or she can move the emitter location via the mouse and get an "immediate" update of the particle path based on the new emitter location. Users benefit greatly from this real-time ability since they can "interact" with the data set. Since the 5.5 release, EnSight has the ability to create and display streamlines in real time as long as the data sets are relatively small.

A significant number of data sets at ARL fall into the "large" (and getting larger) realm. In fact, just to calculate a small number of streamlines in these data sets once can require many seconds, effectively eliminating the ability to produce real-time streamlines.

In the spring of 1995, ARL installed a Silicon Graphics Power Challenge Array containing 8 boxes of 12 R8000 CPU's and 2 GB of memory each (for a total of 96 CPU's and 16 GB of memory), providing a significant opportunity to develop and test parallel applications. With funding from ARL, Computational

Engineering International, Inc., has spent the past few months optimizing and parallelizing the EnSight streamline generator with the hope that the performance gains would be sufficient to allow good interactivity with "large" data sets—a benefit not only to ARL, but to all EnSight users.

3. PERFORMANCE ANALYSIS

In order to maximize the performance of any code, developers should, in the following order, closely examine:

(1) Algorithm	Algorithm modifications have the largest potential for significant performance improvements. If the logic is not sound, performance increases are difficult to gain through other means.
(2) Tuning	Significant performance improvements are possible by paying close attention to the hardware architecture in order to take full advantage of memory and CPU design. Unfortunately, this tuning process usually requires one to have a significant knowledge of the hardware and is thus usually beyond the scope of most application developers.
(3) Parallelization	Significant performance improvements are possible if the application code contains areas that can be performed in parallel. A code can usually be executed in parallel if each processor can work completely independent of any other processor and care is taken to resolve any input/output (I/O) performed in the parallel region.

In our work with EnSight, we have closely examined and implemented modifications to enhance performance through algorithm changes and implementation of parallel regions. Each of these tasks is examined in more detail in the following sections.

4. CHANGES TO THE ALGORITHM

In order to produce the streamlines, the first task is to find an element that contains the user-defined emitter location. We call this task a "global search." Through our own knowledge, and confirmed by using performance monitoring, this global search is the most time-consuming task of the streamline generation and is the bottleneck in gaining performance that allows good interactivity. The algorithm in use had been designed for small memory vector architectures (i.e., Cray), where memory use was ultimately more important than performance (though the performance was not bad on this architecture since the algorithm vectorized). This algorithm did not run well on RISC systems, especially if performance was the goal.

The global search algorithm was completely rewritten with the goal of maximizing performance while keeping the memory usage as conservative as possible. The new algorithm calculates the two bounding largest coordinates of each computational cell, packs the minimum/maximum values into one word of memory, and sorts the lists by the minimum values. A global search using this information can be performed by simply checking for bounding values in each look-up array. Techniques are also implemented to quickly search the arrays and take advantage of the previous search by examining bandwidth parameters.

5. PARALLELIZATION

Restructuring of the streamline algorithm was necessary in order to eliminate memory allocation, memory conflicts, and I/O operations—all of which can cause significant problems in creating parallel streamlines. After this task, we hand-coded the directives necessary to force the parallel region to execute in parallel (the SGI parallel C compiler would not make our do-loop run in parallel because of perceived dependencies due to function calls). As each streamline is calculated, it is communicated to the EnSight client for display. This communication is performed in a "critical" area that allows only one processor at a time to execute. At most, 12 processors are allowed to run the parallel region.

6. RESULTS

In order to test our enhancements, ARL allowed us dedicated use of one of the Power Challenge Array boxes to measure dedicated wall-clock time during testing. In order to eliminate any network load fluctuations that could invalidate our testing, we eliminated all output and display of the streamlines (i.e., we measured only the wall-clock time to compute the traces). All testing was performed with the same data set, using a different number of emitter points and a different number of processors. For comparison, we also modified EnSight 5.5 to eliminate its output and display and tested it with the same data set and emitter points.

The data set tested is shown in Figure 1 and was generated at ARL by Mr. Bernard J. Guidos of the Weapons Technology Directorate, Propulsion and Flight Division, Aerodynamics Branch. The computational grid provided contains 663,000 grid points. The area of interest is in the wake area of the projectile (shown in Figure 2). This wake area has been modeled using two separate data blocks where

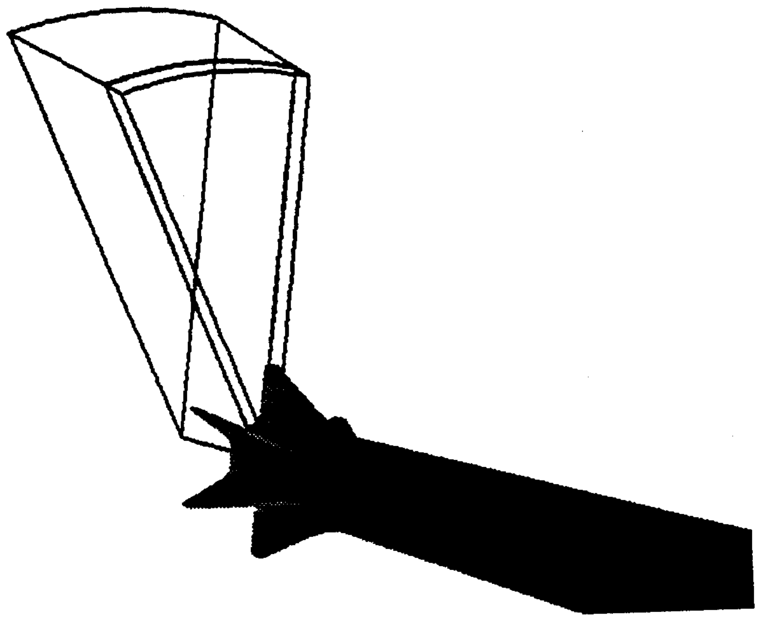


Figure 1. A sample computational grid used to demonstrate interactive particle analysis.

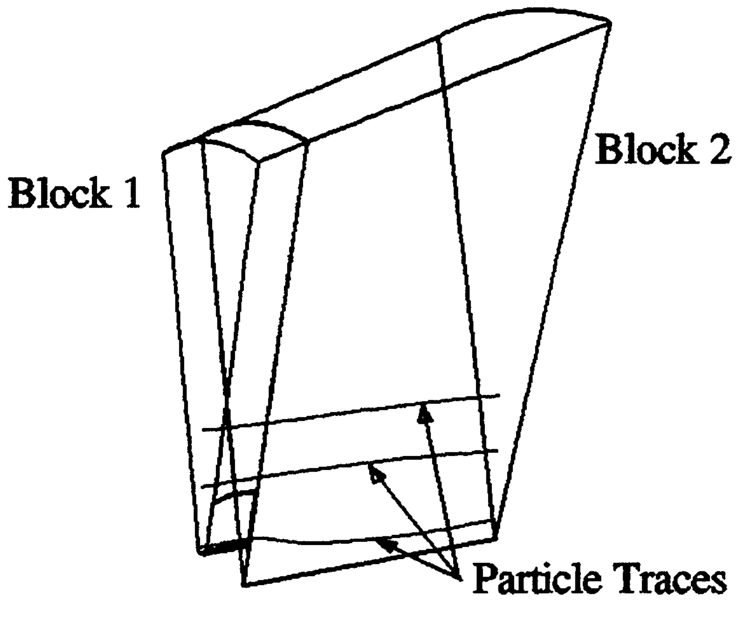


Figure 2. A particle rake in the base flow region of a simplified finned KE projectile.

the mesh density is different between the two blocks. Streamlines are started in the left data blocks as shown and must cross the discontinuity between the two blocks. When the data are converted to the internal EnSight data format, the problem contains 628,170 hexahedral elements.

Table 1 shows the results of varying the number of emission points and number of processors, as well as the timing values for EnSight 5.5. All timings are in wall-clock seconds.

Table 1. Results of Benchmark Testing

No. of Emission Points	Version 5.5 (s)	Version 5.5.1				
		1 CPU's (s)	2 CPU's (s)	4 CPU's (s)	8 CPU's (s)	12 CPU's (s)
3	29.51	0.07	0.05	0.03	0.04	0.04
10	82.68	0.29	0.15	0.15	0.12	0.14
29	205.97	0.96	0.64	0.43	0.34	0.34
75	NA	2.68	1.80	1.28	0.75	0.75
118	915.18	4.97	3.70	2.10	1.73	1.57
173	NA	7.38	5.64	3.39	2.48	2.20
214	1,646.75	8.35	6.23	3.53	2.67	1.88
342	NA	21.47	18.19	13.18	4.90	4.20
1,395	NA	54.72	41.68	26.08	15.35	10.87
3,165	NA	123.50	94.79	58.64	29.94	24.57
5,651	NA	220.60	173.29	104.67	55.02	39.20

NA = Not Attempted

The appendix shows the results in Table 1 in graphical form.

7. ANALYSIS

Each test modifies the number of emission points. As the emission points are not in the same location for each test, care must be taken in drawing direct correlation between tests (i.e., if you double the number of emission points [using one processor], you do not necessarily double the wall-clock time since the streamlines are emitting from different locations in each test). If the mesh and velocity field were completely uniform, these comparisons could be made.

The modified global search algorithm in EnSight 5.5.1 has increased performance by approximately 200 times over the old algorithm as implemented in version 5.5. It must be remembered that this performance increase comes at the expense of approximately 7.5 MB of memory (for a data set of 628k elements). In our opinion, this added memory use is an excellent tradeoff given this substantial performance increase.

Some of the performance gains between EnSight 5.5 and 5.5.1 can probably be attributed to compiler differences. While both versions have been compiled with optimization, version 5.5 was compiled to take advantage of the mips2 instruction set, while the 5.5.1 version tested used mips4.

Adding processors does not always result in a good payoff in terms of performance. The performance gains depend to a large degree on how busy the processors can be kept and over how much time. The best parallel speedup is achieved where there are longer run times overcoming the operating system setup of starting the parallel region. Eight processors give us a speedup of 3–4 times when compared to one processor. To put this in perspective, eight processors give us an approximate performance increase of 600 times when combined with the modified search global algorithm and when compared to the 5.5 release for the test containing 214 emission points. Having more processors than particle traces does not help (see the results for three particle traces).

There appears to be significant overhead of starting a parallel region. This overhead appears to be the same regardless of the number of processors used. This is evident by examining the relatively modest performance increase in going from 1 to 2 processors compared to the performance between 2 and 4, 8, or 12 processors.

The performance increase when adding additional processors is nonlinear. This is probably due to one or more of the following factors (and, possibly, other factors):

(1) Load Balance In order to get a perfect speedup in a parallel region, each processor must do the exact same amount of work (i.e., no processor must be idle). For most applications, this is almost impossible to do, resulting in a load imbalance.

EnSight uses an SGI-specific, load-balancing mechanism called gss (guided self-scheduling), which behaves best when each iteration of the parallel loop varies only slightly from any other iteration. For this particular data set, this is not always true. The streamlines immediately behind the projectile take longer to compute than the rest of the traces—probably significantly longer. This may create a condition in which some of the processors are idle, waiting for others to finish.

(2) Local Searches If a global search has already been performed, EnSight can use the result (the element number containing the emitter) in starting a new streamline by performing a local search. A local search starts at the last known location and uses the neighboring element table and the element shape function to find the new element containing the emitter. This algorithm works great as long as the last location is relatively close to the new location—a condition which is typical if you run on one processor and you are creating several streamlines. However, this condition is probably violated when multiple processors are engaged due to the way the workload is split among the processors.

(3) Memory In a globally shared memory system (such as the SGI), memory contention is a real possibility when utilizing multiple processors. This will cause some of the processors to wait for memory longer than they would if only one processor was being utilized.

One must keep in mind when viewing the results of the performance test that the timings only indicate the time required to generate the streamlines once. When performing streamline generation in a "real-time" environment, the particles are continuously updated as the user moves the emitter location. For example, the test of 29 particle traces would be capable of generating 3 frames/second when using 8 processors, but would only generate 1 frame/second when using 1 processor. In terms of user interactivity, this is a significant difference.

One might be tempted to say that the larger tests, with hundreds or thousands of emitters, are purely academic since no one would want to view the complicated picture which results. However, the curved vector arrow function in EnSight also uses this same algorithm—it is routinely used to calculate curved arrows at the number of emission points used in these tests.

Finally, the results of this test show only a trend, as all of the factors described previously will cause all data sets to behave somewhat differently.

8. CONCLUSIONS

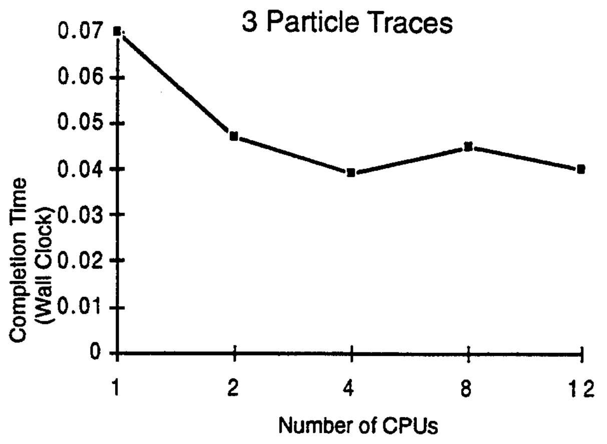
By taking advantage of new algorithms and parallelization, we have shown that real-time particle tracing is possible in data sets typical of those being computed at ARL. The performance increase has been so dramatic that it is not only a difference in performance, but a difference in ability (i.e., we can now process tasks that were impossible in the previous release of EnSight [see preceding tests comparing wall-clock time for 214 emitters]).

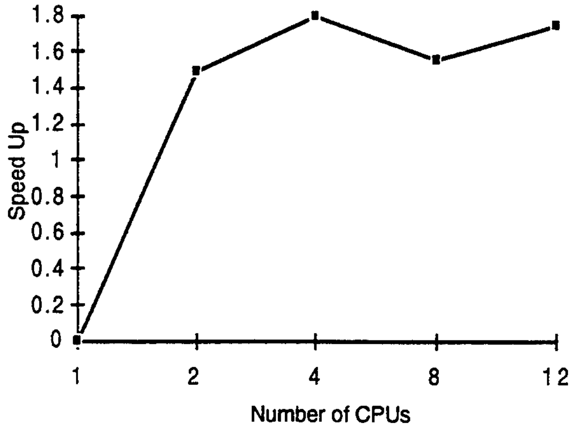
Our experience with this project has shown that there are significant opportunities for performance improvements in EnSight. In the future we hope to expand our work into other EnSight functionalities and possibly investigate performance tuning and code in-lining.

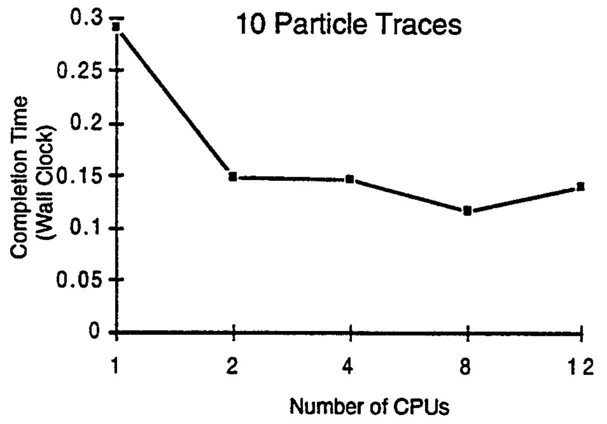
APPENDIX:

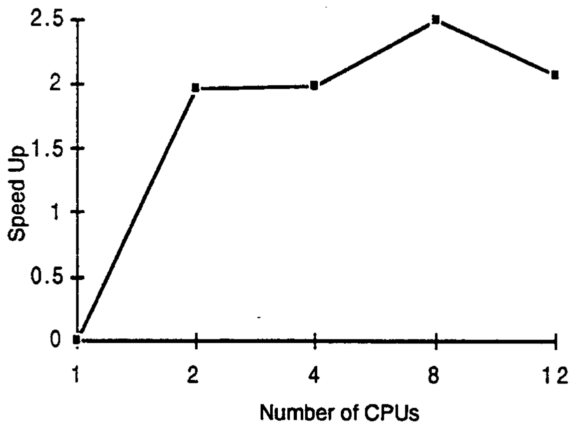
GRAPHICAL RESULTS OF A VARIETY OF BENCHMARK TESTS

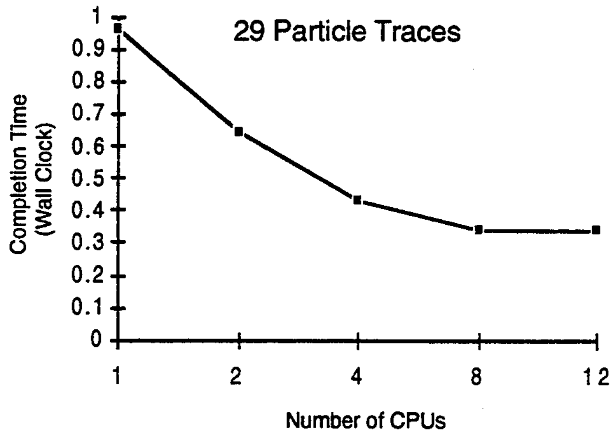
INTENTIONALLY LEFT BLANK.

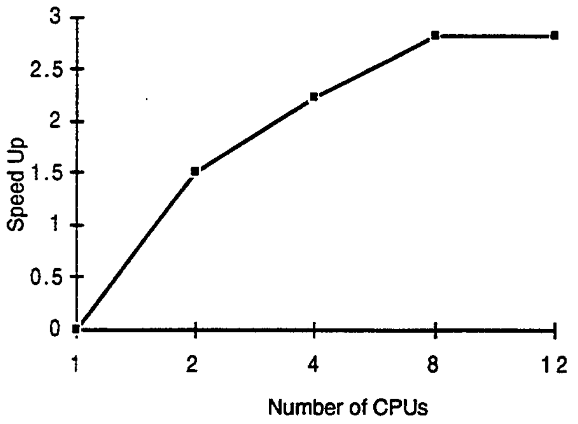

3 Particle Traces
Completion Time (Wall Clock)
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
1
2
4
8
12
Number of CPUs

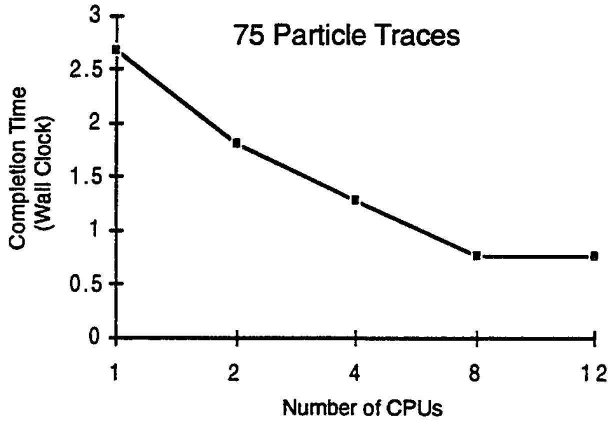

Speed Up
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
1
2
4
8
12
Number of CPUs

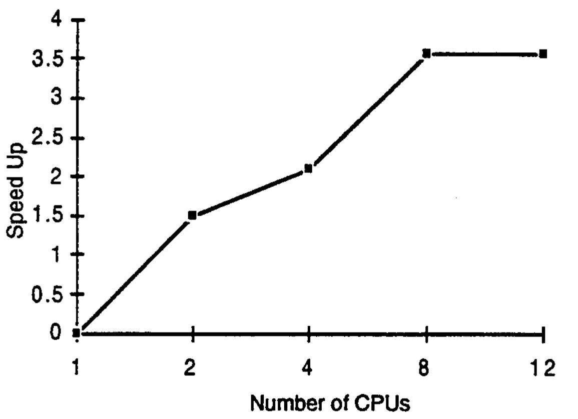

10 Particle Traces
Completion Time (Wall Clock)
0
0.05
0.1
0.15
0.2
0.25
0.3
1
2
4
8
12
Number of CPUs

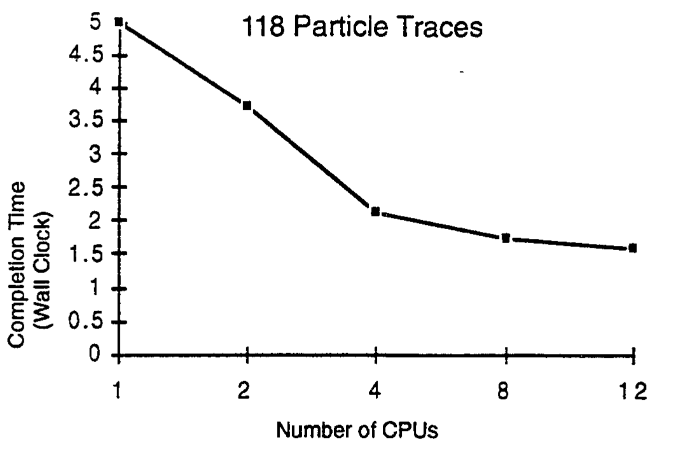

Speed Up
0
0.5
1
1.5
2
2.5
1
2
4
8
12
Number of CPUs

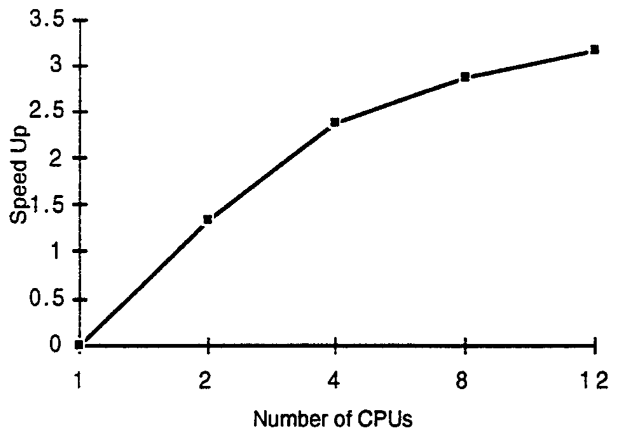

29 Particle Traces
Completion Time (Wall Clock)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1
2
4
8
12
Number of CPUs

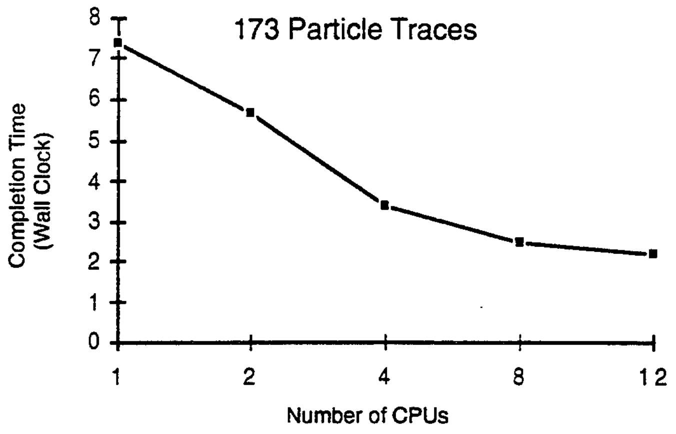

Speed Up
0
0.5
1
1.5
2
2.5
3
1
2
4
8
12
Number of CPUs

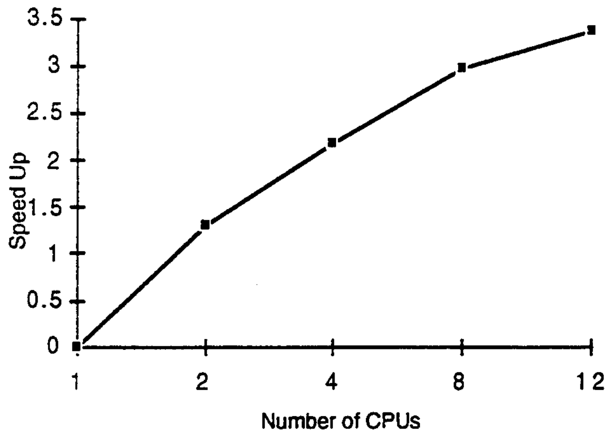

75 Particle Traces
Completion Time (Wall Clock)
0
0.5
1
1.5
2
2.5
3
1
2
4
8
12
Number of CPUs

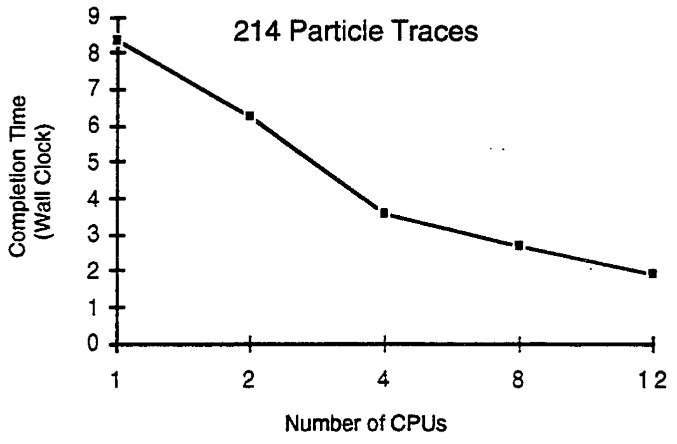

Speed Up
0
0.5
1
1.5
2
2.5
3
3.5
4
1
2
4
8
12
Number of CPUs

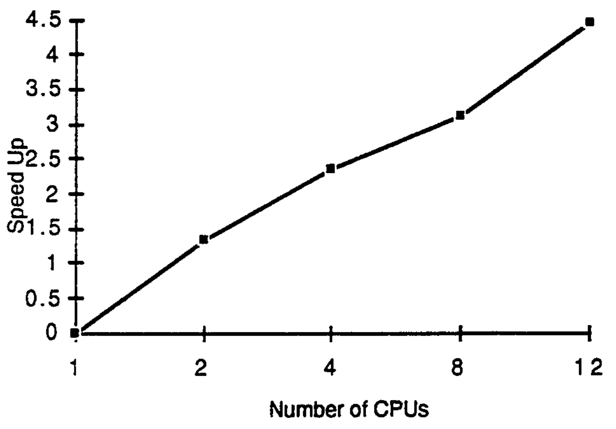

118 Particle Traces
Completion Time (Wall Clock)
Number of CPUs
1
2
4
8
12

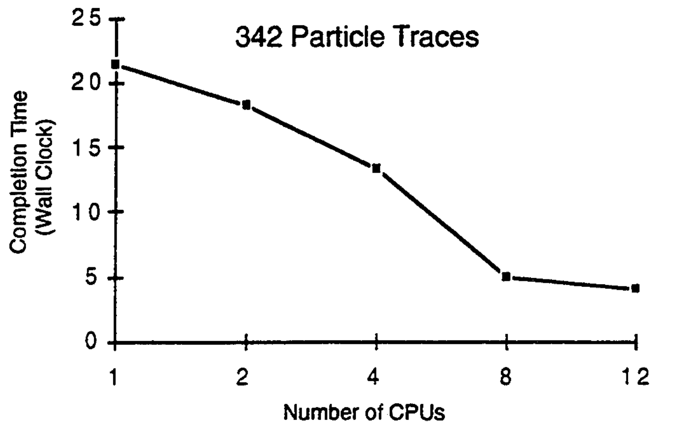

Speed Up
Number of CPUs
1
2
4
8
12

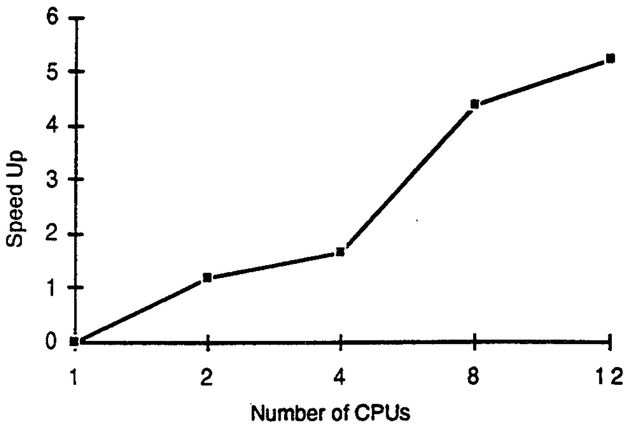

173 Particle Traces
Completion Time (Wall Clock)
Number of CPUs
1
2
4
8
12

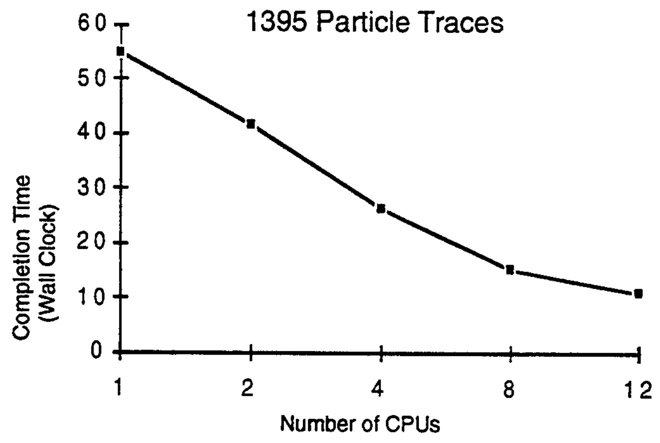

Speed Up
Number of CPUs
1
2
4
8
12

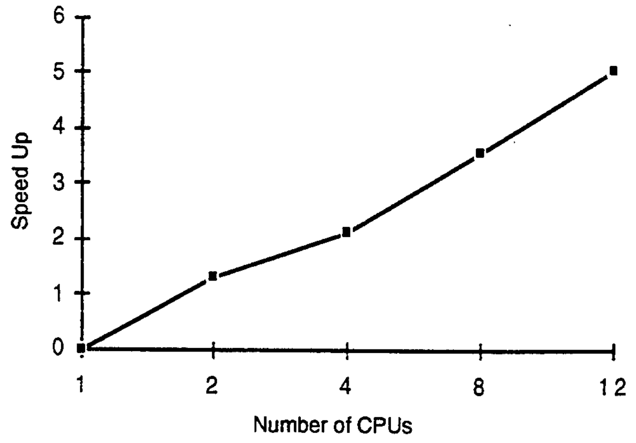

214 Particle Traces
Completion Time (Wall Clock)
Number of CPUs
1
2
4
8
12

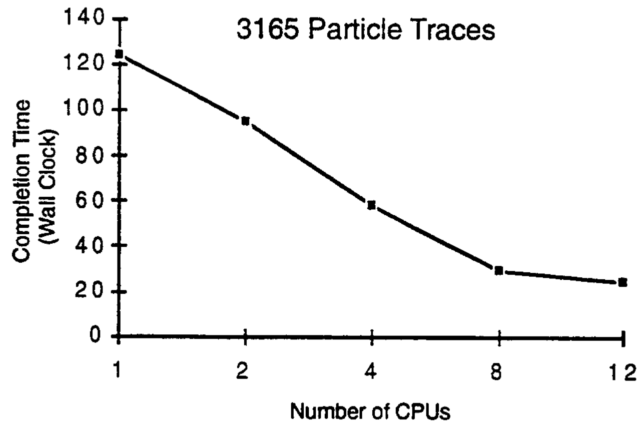

Speed Up
Number of CPUs
1
2
4
8
12

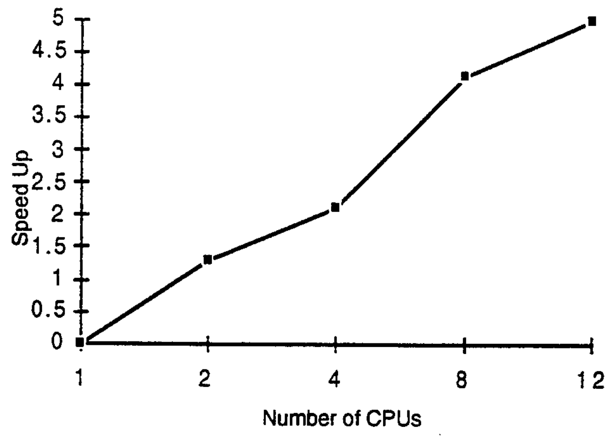

342 Particle Traces
Completion Time (Wall Clock)
Number of CPUs
1
2
4
8
12

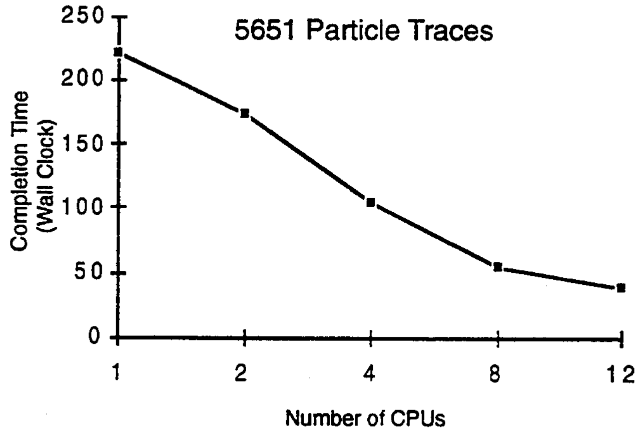

Speed Up
Number of CPUs
1
2
4
8
12

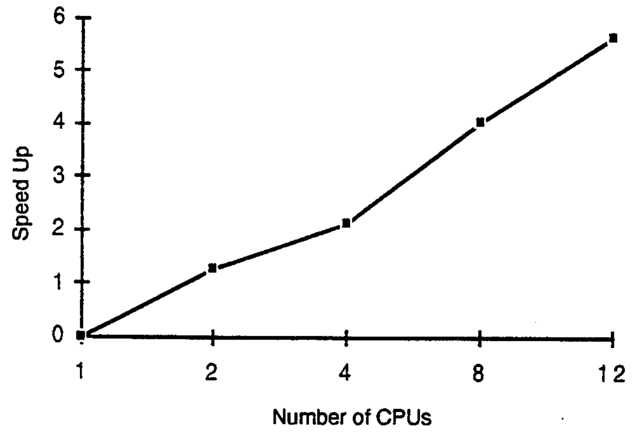

1395 Particle Traces
Completion Time (Wall Clock)
0
10
20
30
40
50
60
1
2
4
8
12
Number of CPUs


Speed Up
0
1
2
3
4
5
6
1
2
4
8
12
Number of CPUs


3165 Particle Traces
Completion Time (Wall Clock)
0
20
40
60
80
100
120
140
1
2
4
8
12
Number of CPUs


Speed Up
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
1
2
4
8
12
Number of CPUs


5651 Particle Traces
Completion Time (Wall Clock)
0
50
100
150
200
250
1
2
4
8
12
Number of CPUs


Speed Up
0
1
2
3
4
5
6
1
2
4
8
12
Number of CPUs

INTENTIONALLY LEFT BLANK.

NO. OF COPIES	ORGANIZATION
2	DEFENSE TECHNICAL INFO CTR ATTN DTIC DDA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TA 2800 POWDER MILL RD ADELPHI MD 20783-1145
3	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TL 2800 POWDER MILL RD ADELPHI MD 20783-1145
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TP 2800 POWDER MILL RD ADELPHI MD 20783-1145
	ABERDEEN PROVING GROUND
2	DIR USARL ATTN AMSRL OP AP L (305)

NO. OF COPIES	ORGANIZATION
2	COMPUTATIONAL ENGRNG INTL ATTN DR ANDERS GRIMSRUD PO BOX 14306 RESEARCH TRIANGLE PARK NC 27709
	ABERDEEN PROVING GROUND
12	DIR, USARL ATTN: AMSRL-SC, W. MERMAGEN AMSRL-SC-C, C. NIETUBICZ M. WILLIAMS AMSRL-SC-CC, D. THOMPSON R. ANGELINI J. HARE J. GROSH AMSRL-WT-PB, B. GUIDOS H. EDGE E. FERRY K. HEAVEY AMSRL-WT-PD, D. HOPKINS

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number/Author ARL-CR-298 (Grimsrud) Date of Report August 1996

2. Date Report Received ______

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) ______

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) ______

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. ______

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) ______

CURRENT ADDRESS

Organization ______

Name ______

Street or P.O. Box No. ______

City, State, Zip Code ______

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

Organization ______

Name ______

Street or P.O. Box No. ______

City, State, Zip Code ______

(Remove this sheet, fold as indicated, tape closed, and mail.)
(DO NOT STAPLE)

DEPARTMENT OF THE ARMY

OFFICIAL BUSINESS

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 0001,APG,MD

POSTAGE WILL BE PAID BY ADDRESSEE

DIRECTOR
U.S. ARMY RESEARCH LABORATORY
ATTN: AMSRL-SC-CC
ABERDEEN PROVING GROUND, MD 21005-5067