INTRODUCTION

- As the size of problems to which HPC can be applied continues to increase, so does the processing demand. Countries are competing fiercely to build the world’s fastest supercomputer, similar to the Space Race in the mid-to-late 20th century.
- To efficiently employ such large and expensive systems, developers must study application behaviors by collecting detailed performance information with the help of performance tracing tools.
- Tracing toolsets could severely affect the performance and scalability of the running application:
  - Scalability: Due to the large I/O ratio that must be computed for modern large-scale machines, collecting all detailed performance information may not be feasible from a scalability perspective
  - Performance: The tracing processes may compete with the application for resources, which can perturb the application’s behavior
- The overarching objective of this project is to develop a scalable tracing toolset that can cluster processes with the same behaviors into groups quickly and with low overhead; then, instead of collecting performance information from all individuals, it can collect the information from just a set of representatives.
- We applied our clustering algorithm on trace files created by ScalaTrace, an MPI tracing toolset that provides orders of magnitude smaller, if not near-constant size, communication traces regardless of the number of nodes, while preserving structural information.
- ScalaTrace has two-stage trace compression:
  1. Intra-Compression:
     - Extended Regular Section Descriptors (RSDs) capture the loop structures of one or multiple communication events. Power-RSDs (PRSDs) recursively specify RSDs in nested loops.
  2. Inter-Compression:
     - After each node has created its own compressed trace file and the program is completing, ScalaTrace performs an inter-node compression over a radix tree rooted in rank 0. During this reduction, internal nodes combine their traces with other task-level traces received from child nodes.

A NOVEL CLUSTERING ALGORITHM

- While intra-compression is fast and efficient, inter-compression is a costly operation that is $O(n^2 \log P)$, where $n$ is the number of MPI events and $P$ is the number of processes. We proposed and applied a signature-based clustering algorithm to reduce the bottleneck effect.

THE EXPERIMENTAL RESULTS

- To evaluate the accuracy and scalability of our algorithm, we created Reference Clustering which is the un-aggregated version of Call-path, Parameter Clustering.
- We utilized ARC cluster computer. All machines were 2-way SMPs with AMD Opteron 6128 processors, 8 cores per socket.

CONCLUSIONS

- For Exascale computing, it is appropriate to split the merge process and to have $\log P$ time complexity and low overhead at the clustering level.
- Unlike CLARA [1] and CAPEK [2], our algorithm is based on exact matching, not on a random sampling process that may reduce accuracy

ACKNOWLEDGEMENTS

- This research was supported by the National Science Foundation, award number 1217748.