Scalability of Sparse Matrix Dense Vector Multiplication

Brian Page; Peter Kogge PhD.
University of Notre Dame

Abstract

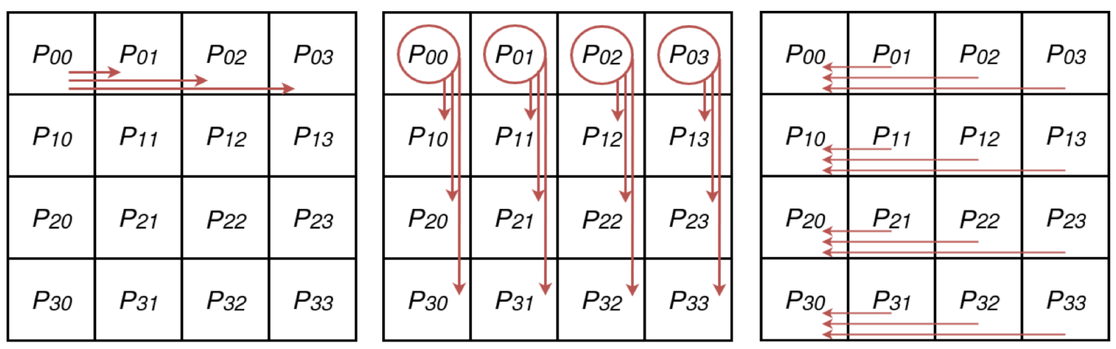
Sparse algebra problems are an integral component of many scientific computing applications. Sparse operations pose a key issue along the path to exascale systems. Hybrid codes seek to capitalize on higher levels of parallelism, while benefiting from a potential for reduced global communication constructs.

Much work has been performed with the intent at reducing SpMV computation time, however in an a distributed environment SpMV is not compute bound, nor is it memory bandwidth bound. We seek to show that utilizing existing traditional HPC computing methodologies constitutes the primary bottleneck in distributed SpMV and perhaps other sparse algebra problems as well.

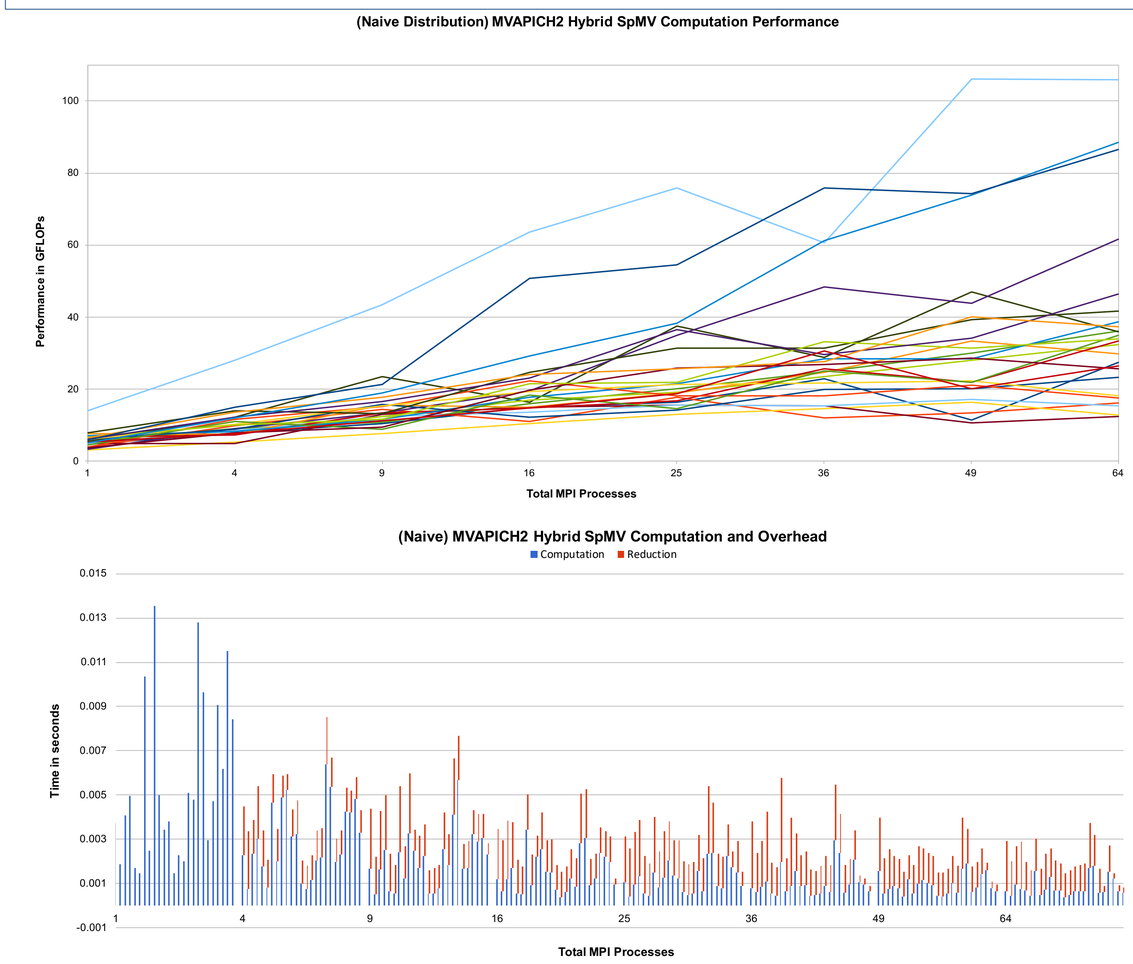
Problem

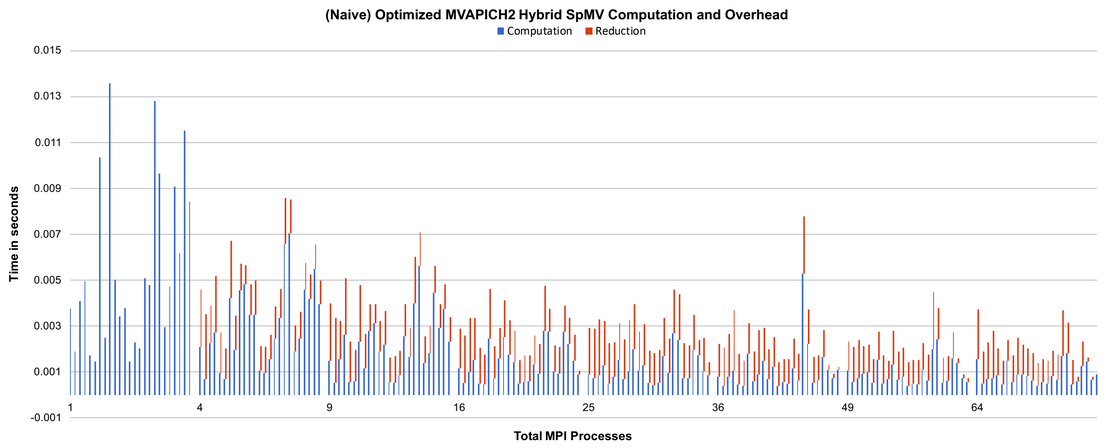
- Sparse Problems do not scale well.
- Attempts to distribute data amongst MPI processes can create load imbalance
- Performance can be come dependent on matrix characteristics
- MPI Communication overhead quickly outpaces performance gain from strong scaling.

Naïve Distribution



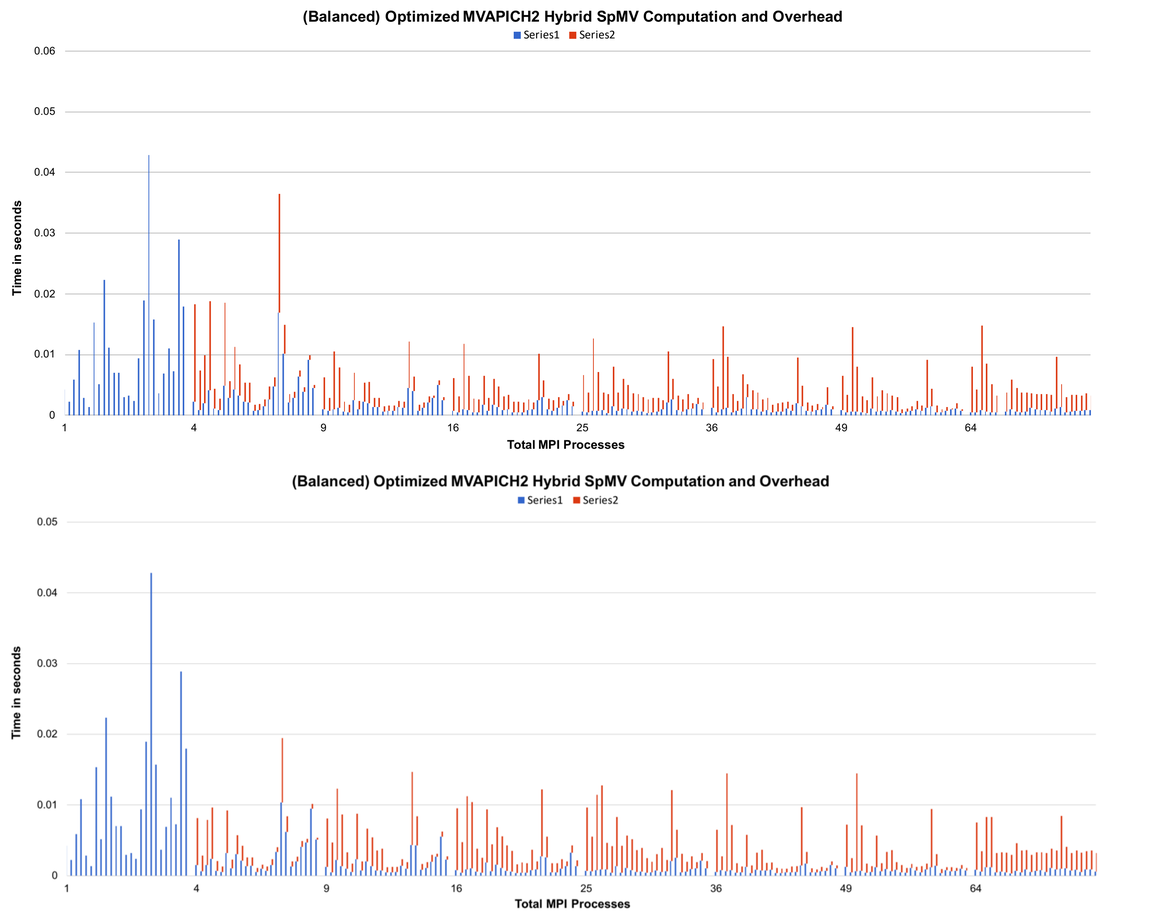
- Input matrix A split into equal size sub matrices, compressed into CSR_i.
- Sub matrix S_{ij} is assigned to the I,j^{th} process within the MPI process matrix P.
- Global master P_{00} sends all data for col_j to col master process P_{0j}.
- Column masters distributed data assigned to individual processes with its column in P
- After SpMV computation, MPI_Reduce with sum collects results for each row portion row_{ij} at the row master process P_{i0}.





Balanced Distribution

- Non-zeros compressed into CSR format, entire contiguous rows sorted by length.
- Greedy bin packing algorithm determines “optimal” non-zero per row distribution.
- Data sent to column masters and to indiduals via similar communication pattern as the naïve method.
- After SpMV computation, initiate MPI_GATHERV on MPI_COMM_WORLD.
- Create near uniform load distribution, while reducing reduction time.



Discussion

$$\frac{2\sqrt{(p)}log_2(\sqrt{(p)})(\frac{m}{\sqrt{(p)}}d+\phi)}{B} \quad \frac{plog2(p)(md+\phi)}{B} \quad \frac{md+(p-1)\phi}{B}$$

Conclusions

It can be seen that the MPI communication time far exceeds the useful computation portion of the program, despite our best attempts to reduce such overhead.

The equations seen above represent the MPI overhead as a function of process count and matrix rows and non-zeros. In all cases, including the ***oracle,*** as p increases the number of non-zeros per row decreases, computational performance flattens out as communication becomes the primary factor driving process/node work load.

Contact

Brian Page
University of Notre Dame
Email: bpage1@nd.edu

© Poster Template by Genigraphics® 1.800.790.4001 www.genigraphics.com