Parallel POPCycle Implementation

Ben Steffen- Integrative Biology

CS267: Applications of Parallel Computers-Spring 2009

Abstract:

This project involved implementing a parallel version of POPCYCLE, an individual-based model of the population dynamics of marine zooplankton. The model is parallelized by having a master process subdivide the population among slave processes, with each slave process computing for its subpopulation independently and then reporting back to the master process at set intervals. This parallel implementation achieves significant speedups over the serial version, particularly when the population size is large. Increasing the frequency of communications between the master and slaves slows the model slightly, but is not a major performance bottleneck. The results also suggest that to attain additional speedup of the algorithm, it may be necessary to look at the file output process.

Background:

POPCYCLE is an individual-based simulation model for the population dynamics of marine zooplankton. The model tracks a population of individuals through growth, reproduction, mortality, and tracks where individuals move and are advected within a realistic flow field. The physical and biological fields used by POPCYCLE are generated by another model, the Regional Ocean Modeling System (ROMS).

Figure 1.

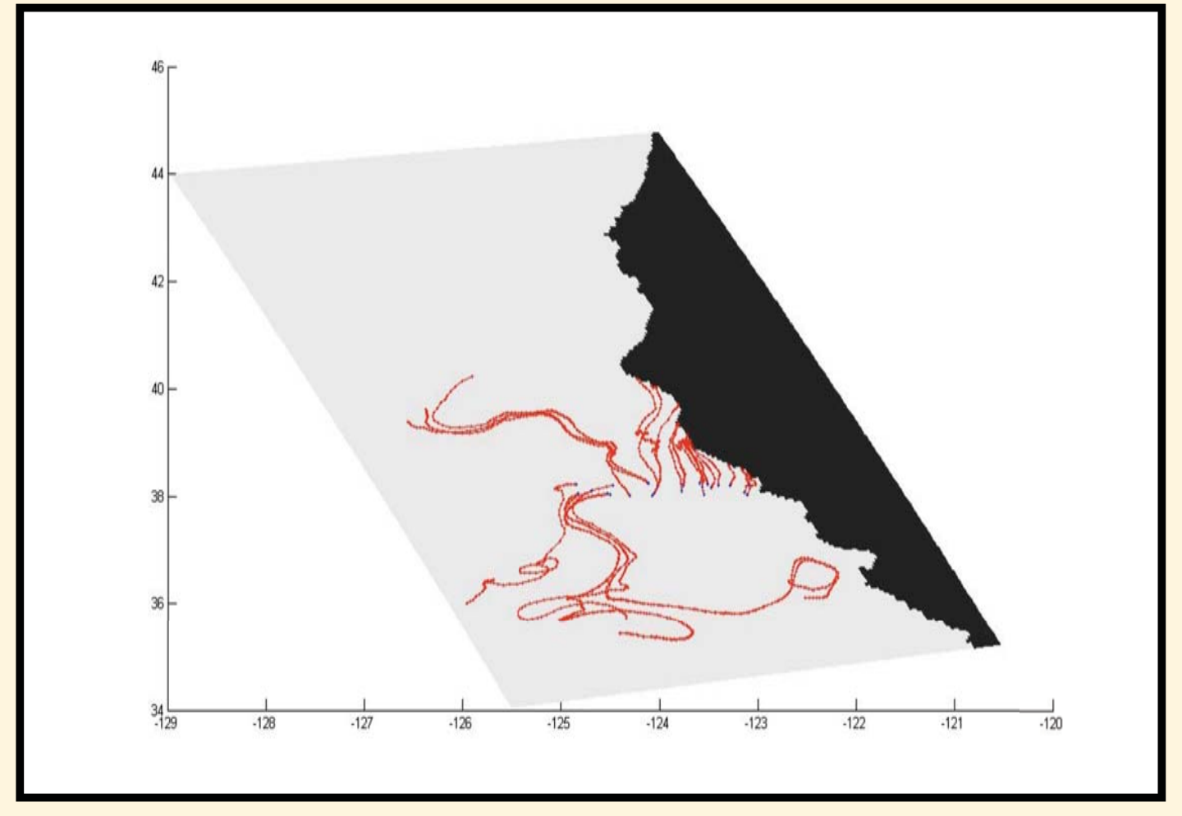


Figure 2

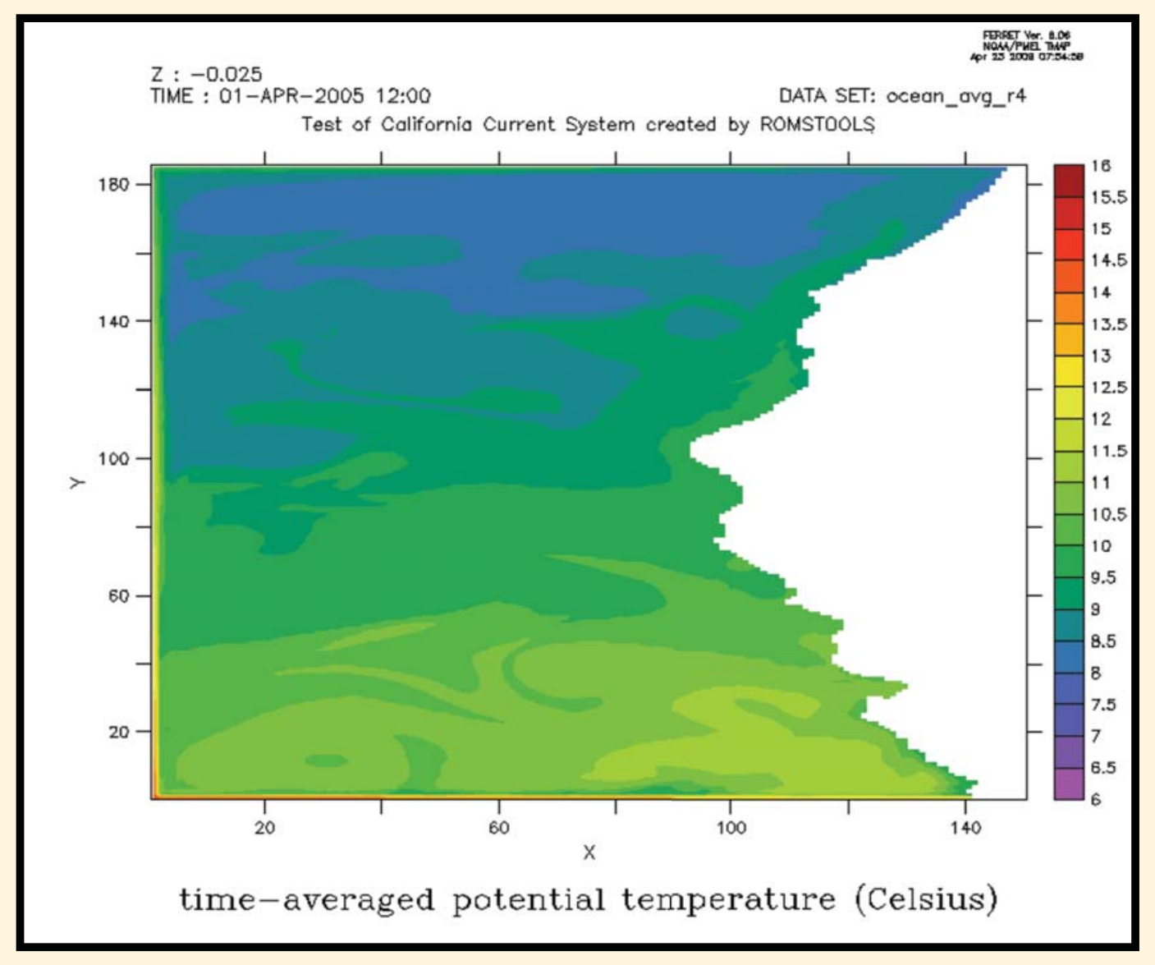


Figure 1. Plot of spatial data produced by POPCYCLE over a several-day run.

Figure 2. Sample of sea-surface temperatures for the California Current produced by the Regional Ocean Modeling System (ROMS).

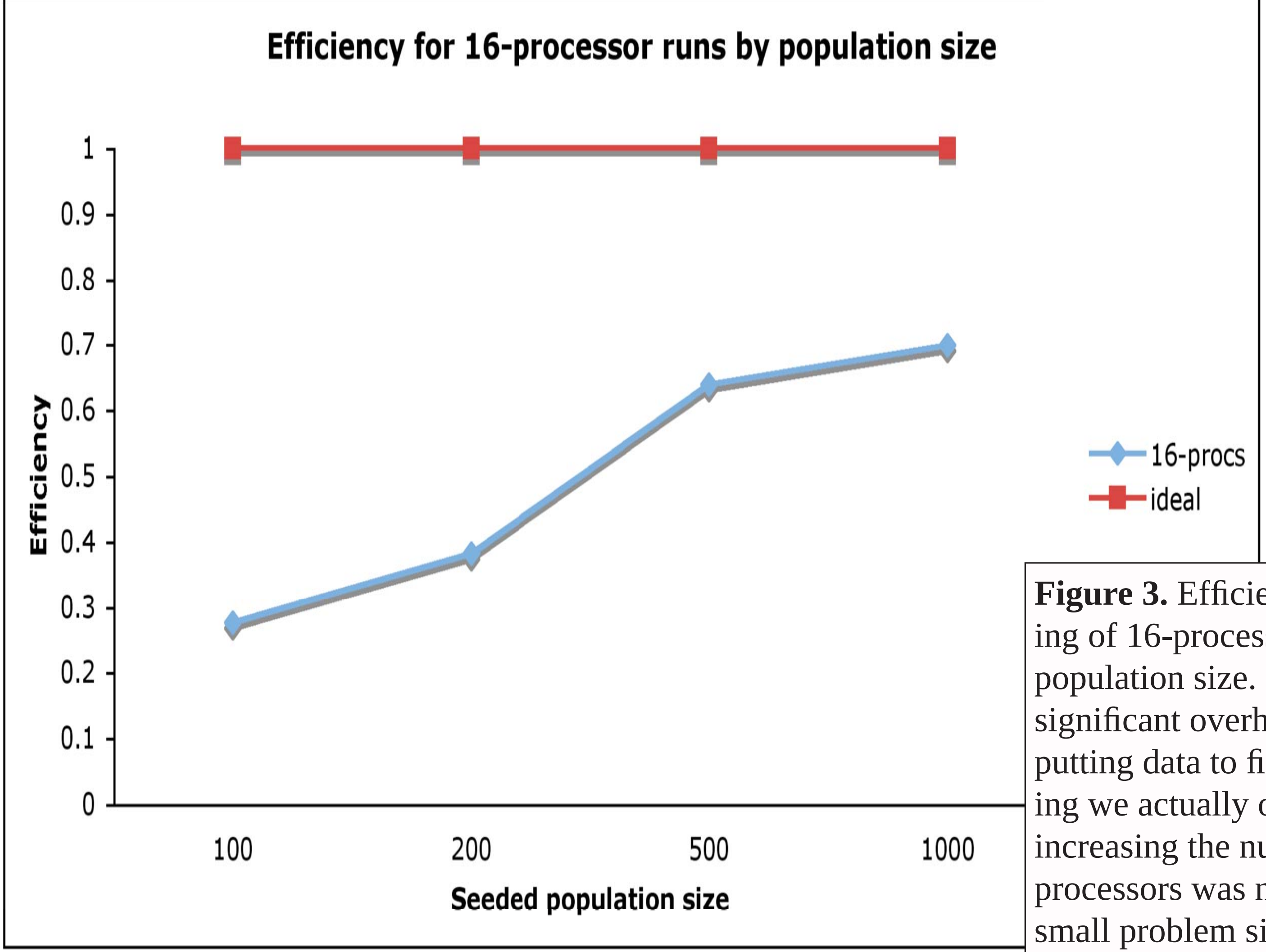
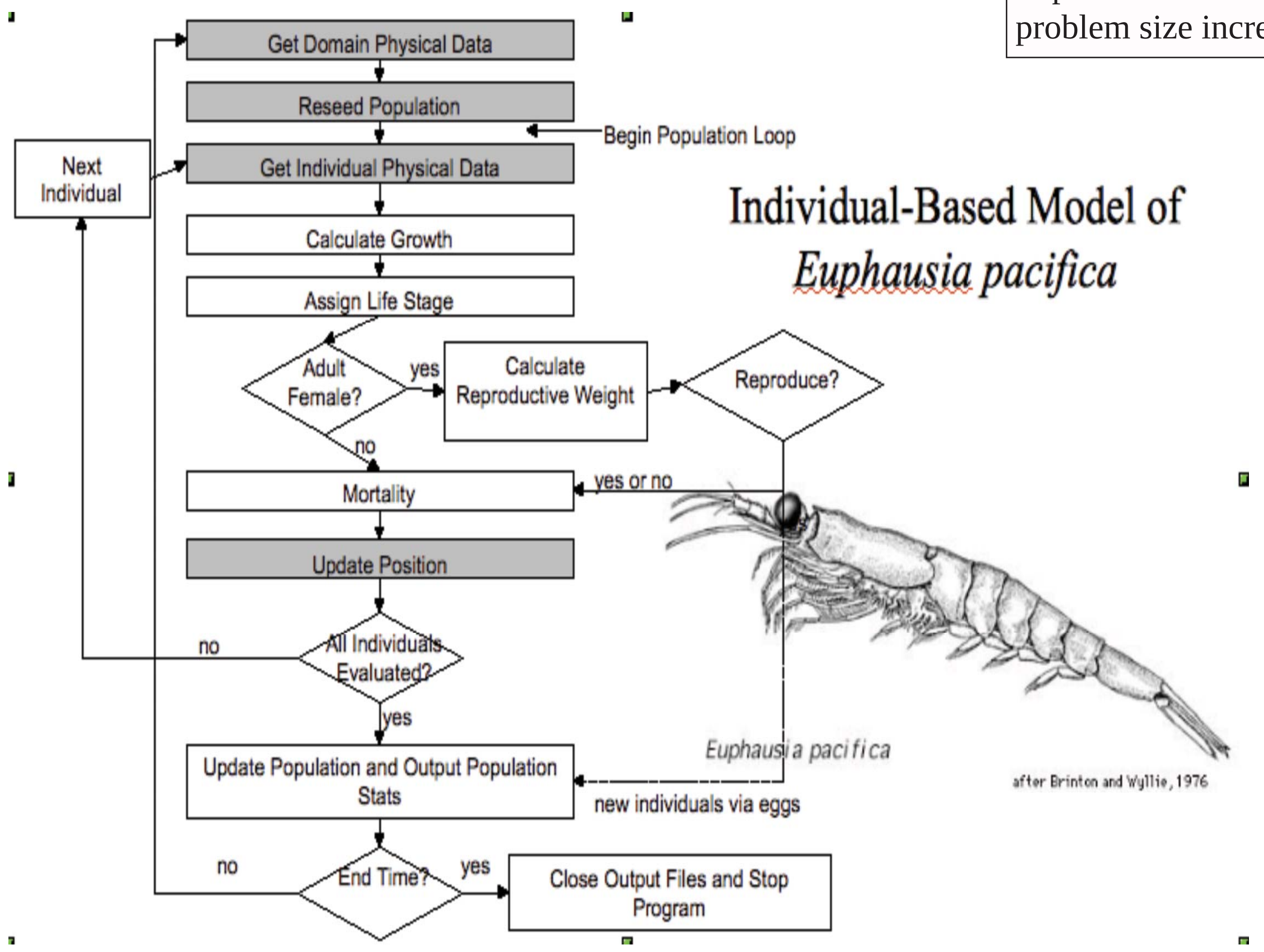


Figure 3. Efficiency scaling of 16-processor runs with population size. Due to the significant overhead of outputting data to files, the scaling we actually obtained for increasing the number of processors was not great for small problem sizes, though it improved substantially as the problem size increased.



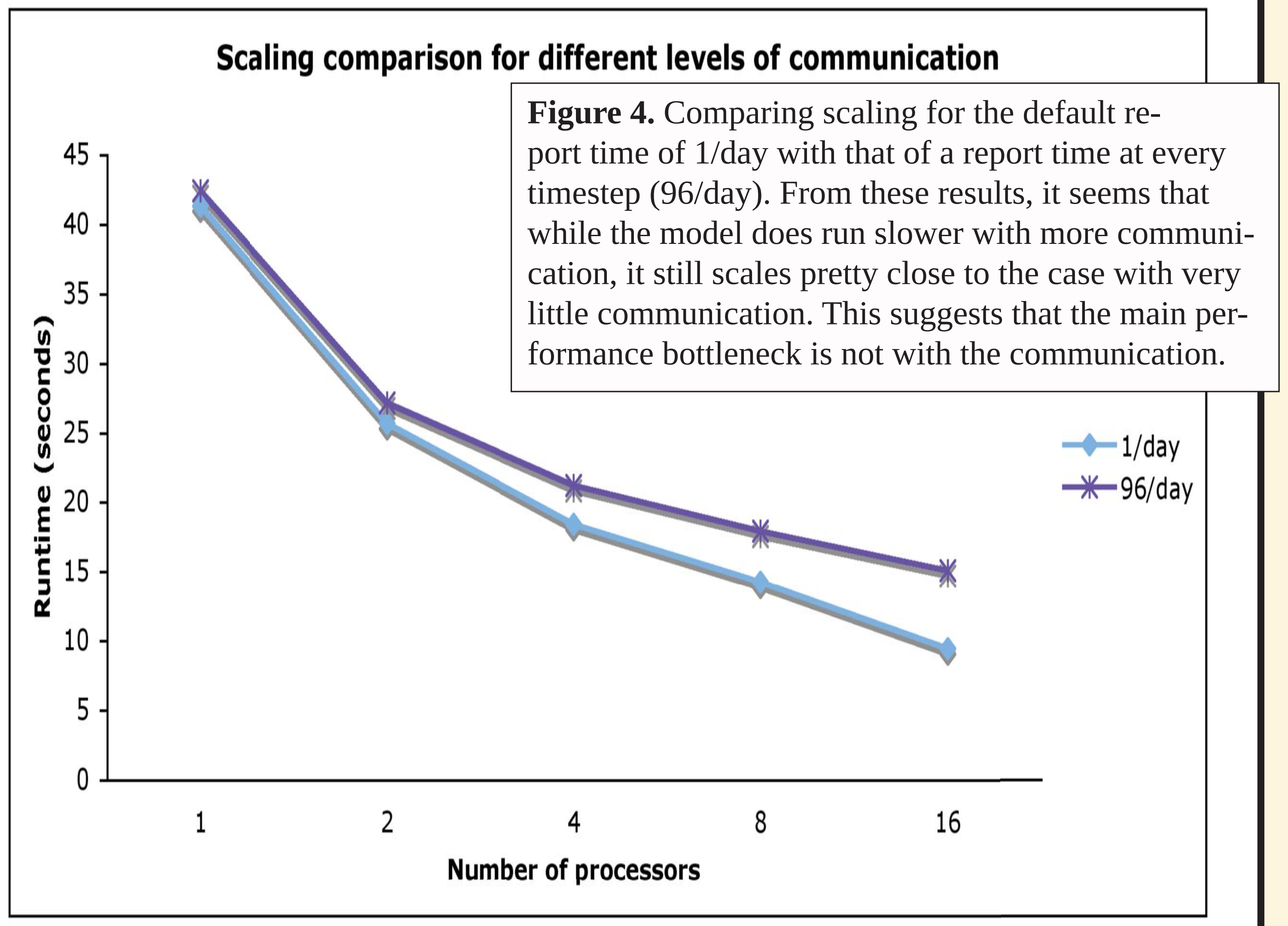


Figure 4. Comparing scaling for the default report time of 1/day with that of a report time at every timestep (96/day). From these results, it seems that while the model does run slower with more communication, it still scales pretty close to the case with very little communication. This suggests that the main performance bottleneck is not with the communication.

Parallel implementation:

The motive for working on the parallel implementation of POPCYCLE was to set up a parallel framework on which future versions could be built. The basic design for the parallel implementation of POPCYCLE was to have a master process distribute the work to slave processes. Each slave process receives its own segment of the population from the master, and computes time steps for that subpopulation up until a fixed 'report' time, when all slave processes send their subpopulations back to the master, which allows the master to output statistics on the entire population before redistributing the population. The particles are not split up according to their spatial location, so each slave process does not need to keep track of when to pass individuals to other processes. This way of dividing up the population would have to change if there were interactions between neighboring individuals, but for the model's current application to modeling krill, it is unlikely that there will be any interaction between individuals other than through the density of the entire population.

Conclusion:

As expected, implementing a parallel version of POPCYCLE does yield a significant speedup in time to solution. Scaling on small initial population sizes is less than ideal, but improves greatly for larger populations. In practice, the poor scaling for small populations should not be a problem, as the parallel solution will only be needed for large problem sizes. The fact that communication did not seem to be a major bottleneck will be helpful if neighbor interactions are ever implemented, which will involve a more communication intensive model. Lastly, the issue with the file output possibly slowing the model may have to be dealt with, but the format of the output is likely to change with specific applications of POPCYCLE, so it may be more suitable to address that problem then.