Performance Analysis of DroughtHPC

Yasodhadevi Nachimuthu
Portland State University, yasodhan@pdx.edu
Performance Analysis of DroughtHPC

Yasodha Nachimuthu

Department of Computer Science

Faculty advisor: Dr. Karen L. Karavanic, Department of Computer Science

Project Principal Investigator: Dr. Hamid Moradkhani, Department of Civil and Environmental Engineering

Motivation

• Analyze the performance and identify the runtime bottlenecks of DroughtHPC [1], a drought prediction application developed at PSU
• Scale the application to do fine-grained simulations, and simulate a larger geographical area
• Obtain a holistic view of the application, to use resources effectively

Background - DroughtHPC

• Predicts drought for a target geographical area
• Uses variety of data: soil conditions, canopy cover, snow accumulation, vegetation layers, and meteorological data of the target area
• Application is written in Python, and uses two hydrologic models VIC [2] and PRMS [3]
• Target geographical area is divided into jobs, with group of 25 cells in each job
• Parallel computing is used to simulate jobs concurrently

Methodology

• Evaluate single-cell simulation performance first, to identify major runtime contributors
  • Analyze the timing and memory footprint of hydrologic models
  • For single-node performance, study the correlation between runtime and the problem size (number of jobs that can be simulated concurrently)
  • For multi-node performance in a Linux cluster, analyze the runtime and effects of interference from other processes running in the cluster
• Tools used – Python cProfile, GPROF [4], Valgrind

Example of Python cProfile log

```plaintext
42508685 function calls (39864755 primitive calls) in 5763.898 seconds
Ordered by: cumulative time
ncalls  tottime  percall  cumtime  percall  filename:lineno(function)
576/1    0.035    0.000    5764.093  5764.093  {built-in method exec}
1       207.624  207.624  5764.093  5764.093  data_assimilation.py:1(<module>)
36600   0.759    0.000  5346.668  0.146  subprocess.py:919(communicate)
36600   10.151   0.000  5345.835  0.146  subprocess.py:1575(_communicate)
```

Conclusions and Future work

• We identified major performance bottlenecks:
  • For single-cell simulations, VIC hydrology model along with Python subprocess call
  • For multi-node simulations, using NFS with MPI
• We concluded that the best fit for problem size on a single-node equals the number of cores
• We are investigating increasing parallelism using OpenMP, XeonPhi, and GPU technologies
• We are developing new performance tools to obtain the holistic view of the application

References