Performance Analysis of DroughtHPC and Holistic HPC Workflows

Yasodha Suriyakumar yasodhan@pdx.edu
Karen L. Karavanic karavan@pdx.edu
Hamid Moradkhani hmoradkhani@ua.edu
Portland State UNIVERSITY

Goals

- Analyze the performance of DroughtHPC [1], a drought prediction application developed at Portland State University
  - DroughtHPC improves prediction accuracy for a target geographical area; uses data assimilation techniques that integrate data from hydrologic models, and satellite data
  - Uses variety of data: soil conditions, snow accumulation, vegetation layers, canopy cover and meteorological data
- Scale the application to do finer-grained simulations, and to simulate a larger geographical area

Implementation

- Application is written in Python, and uses two hydrologic models VIC [2] written in C, and PRMS [3] written in FORTRAN and C
- Land surface of the target geographical area is modelled as grid of uniform cells, and simulation divides it into jobs, with group of 25 cells in each job
- For a job that simulates 50 meteorological samples and one month time period, input data size is 144.5 Mb, with the satellite data consuming 132 Mb
- Runtime for a job on single-node is approximately two hours with the initial Python prototype

Methodology

Evaluate sequential single-cell simulation performance
Analyze timing and memory footprint of hydrology models

↓

For parallel single-node performance, study the correlation between runtime and the problem size

↓

For parallel multi-node performance in a Linux cluster, analyze effects of interference from other processes

Results: Single Node

Runtime data of DroughtHPC (with VIC) for 50 meteorological samples and one month simulation on a single node (8 cores)

Number of jobs (group of 25 cells)	Minimum runtime (hours: minutes)	Maximum runtime (hours: minutes)
1	2:21	2:21
2	1:54	1:55
4	1:49	1:50
8	1:48	1:50
12	2:42	2:44

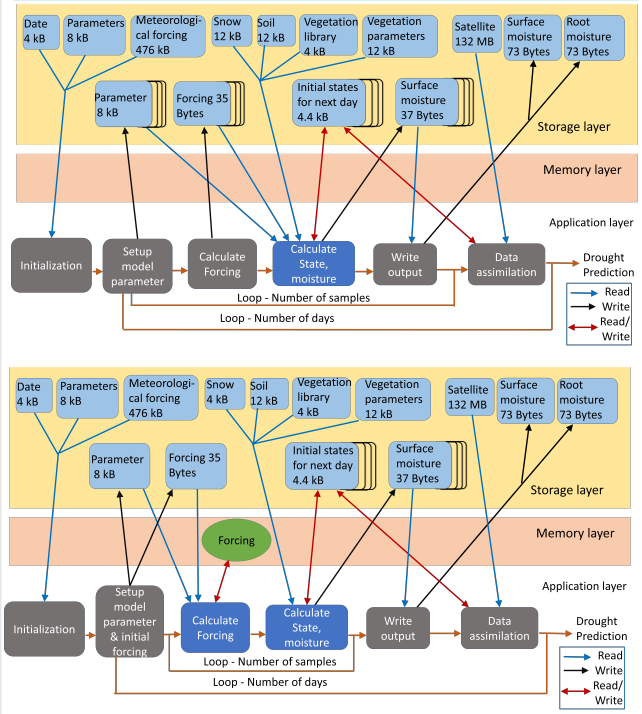
Observations

- No single performance tool characterizes the calling pattern / interactions between Python and VIC
- Profiling does not focus attention to the number of files or frequency of file creation/access
- Manual integration of data from multiple performance tools is time-consuming: Python cProfile, Valgrind, Strace system utility, etc.

Summary and Future Work

- Single-cell simulations: bottleneck is the overhead of the VIC hydrologic model call from Python
- Parallel single-node performance:
  - The best fit on our platform is one job per logical core;
  - We explored changes to VIC for Intel Xeon Phi
  - We are developing a version of VIC that eases integration with individual science codes such as data assimilation
  - Multi-node simulations with MPI: bottleneck is the filesystem access pattern
- We designed PPerfG for visualizing Holistic HPC Workflows
- We implemented a prototype of PPerfG

Holistic HPC Workflows: DroughtHPC



2 Holistic workflow diagrams of DroughtHPC: Files accessed in simulation of a single cell, with multiple meteorological data (Number of samples), and time period (Number of days). Gray sections refer to Python code, and blue refers to the hydrologic model . Top: VIC accesses 57 files in total in each call. Bottom: Hydrologic model (VIC) code is changed to minimize forcing data's movement between storage layers, and reduce invocation cost

PPerfG

- PPerfG: A Visualization Tool for Holistic HPC Workflows for use in performance diagnosis
- Captures the data movement behavior between storage layers, and between different stages of an application
- Challenges: Determining best metrics, and efficient measurement techniques
- Status: initial prototype developed

References

[1] H. Yan and H. Moradkhani, "Combined assimilation of streamflow and satellite soil moisture with the particle filter and geostatistical modeling", *Advances in Water Resources*, vol. 94, pp. 364-378, 2016.
[2] University of Washington, "VIC hydrology model", http://www.hydro.washington.edu/Lettenmaier/Models/VIC/index-old.shtml, 2014. Accessed: 2018-02-01.
[3] U.S. Department of the Interior - U.S. Geological Survey, "Precipitation Runoff Modeling System", https://wwwbrr.cr.usgs.gov/projects/SW_MoWS/PRMS.html , 2016. Accessed: 2018-02-01.
[4] S. L. Graham, P. B. Kessler, and M. K. McKusick, "Gprof: A call graph execution profiler," *SIGPLAN Not.*, vol. 39, pp. 49 - 57, Apr. 2004.
[5] Karen L. Karavanic, John May, Kathryn Mohror, Brian Miller, Kevin Huck, Rashawn Knapp, and Brian Pugh, "Integrating Database Technology with Comparison-based Parallel Performance Diagnosis: The PerfTrack Performance Experiment Management Tool", *In Proceedings of the 2005 ACM/IEEE conference on Supercomputing (SC '05)*.

Acknowledgements

Henry Cooney, Tu Le, and Jiaqi Luo contributed ideas, discussions and implemented software used in this project. This material is based upon work supported by the National Science Foundation under Grant No.1539605. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.