Development of a Knowledge-Sharing Parallel Computing Approach for Calibrating Distributed Watershed Hydrologic Models

Abstract: A research gap in calibrating distributed watershed hydrologic models lies in the development of calibration frameworks adaptable to increasing complexity of hydrologic models. Parallel computing is a promising approach to address this gap. However, parallel calibration approaches should be fault-tolerant, portable, and easy to implement with minimum communication overhead for fast knowledge sharing between parallel nodes. Accordingly, we developed a knowledge-sharing parallel calibration approach using Chapel programming language, with which we implemented the Parallel Dynamically Dimensioned Search (DDS) algorithm by adopting multiple perturbation factors and parallel dynamic searching strategies to keep a balance between exploration and exploitation of the search space. Our results showed that this approach achieved super-linear speedup and parallel efficiency above 75%. In addition, our approach has a low communication overhead, along with the positive impact of knowledge-sharing in the convergence behavior of the parallel DDS algorithm.

Due to the extensive spatiotemporal heterogeneity of predominant hydrological processes, distributed watershed hydrologic models require numerous model parameters and input datasets (Singh and Frevert, 2003; Khakbaz et al., 2012; Refsgaard, 1997). One distributed model simulation accounts for the most time-consuming single computation in watershed modelling, with a runtime varying from minutes to days (Zhang et al., 2009; Razavi et al., 2010). Subsequently, applying global (meta)heuristic optimization algorithms imposes high computational requirements on calibrating these hydrologic models (Xu et al. 2013). The calibration requires an abundant number of model simulations for thorough exploration and exploitation of their hyper-dimensional parameter search space. This substantial computational burden can also get exacerbated by using high-resolution datasets and/or working on larger watersheds (Budamala and Mahindrakar, 2021). Thus, not only the calibration of distributed watershed hydrologic models is a computationally expensive process, but also its complexity and size can potentially grow.

Parallel computing is the simultaneous execution of independent and parallelizable components of a large computing application. This approach is promising to provide speedup (the ratio of execution times of a parallel application to its sequential version) and scalability (the ratio of actual speedup to the ideal speedup with a certain number of processors) for watershed hydrologic model calibration with any size or complexity (Asgari et al., 2022; Quinn, 1994). Sub-model and model level parallelization are two popular parallelization strategies. At the sub-model level, parallelization takes place for the sub-units of a hydrologic model to speed up a single-model simulation. However, it comes with challenges of model reconstruction and complexities associated with parallel task management for component communications (Lin and Zhang, 2021). The model level parallelization is to parallelize integrated model simulations to speed up the *entire* model calibration process, which has shown much ease in achieving parallel computing objectives. For instance, Xia et al. (2021) developed Parallel Optimization with Dynamic Coordinate Search using Surrogates (PODS) to not only decrease the number of model simulations but also increase the calibration speed. The implemented parallelization strategy was at the model level, and the authors reported speedy automatic calibration in their study. Open Multi-Processing (OpenMP) (Huo et al., 2018; Kan et al., 2018; Kan et al., 2020; Kan et al., 2016) and Message Passing Interface (MPI) (Li et al., 2015; Vrugt et al., 2006; Kim and Ryu, 2019) are two popular parallelization paradigms for model level parallelization strategy in the calibration of watershed hydrologic models. However, the current research gap in the parallel calibration of hydrologic models challenges the applicability of these paradigms.

Population-based metaheuristic algorithms, such as Artificial Bee Colony (ABC) (Huo et al., 2018), Bayesian optimization (BO) (Ma et al., 2022), shuffled complex evolution (SCE-UA) (Kan et al., 2018), particle swarm algorithm (PSO) (Ma et al., 2021; Niu et al., 2021), and Genetic Algorithm (Zhang et al., 2016) have been extensively implemented in parallel calibration of hydrologic models. Contrary to single-solution algorithms, these algorithms search the parameter space with multiple initial/evolved solutions. This behavior empowers population-based metaheuristic algorithms to investigate different zones simultaneously and eliminate the bias in the initial solution or sampling strategy(Prügel-Bennett, 2010). Hence, they are inherently parallelizable as the initial population can be split into subpopulations to get evolved independently on parallel computing units. However, algorithms such as ABC and PSO are *swarm-based* algorithms; the evaluation of the population in these algorithms has a cooperation phase in which individuals transfer the information they have gained to lead each other to the global optimum (Ab Wahab et al., 2015; Beheshti and Shamsuddin, 2013). Although splitting the populations into subpopulations increases the diversity and lowers the possibility of getting trapped in a local optimum, it eliminates the cooperation phase and impacts the convergence speed. In addition, Hadka and Reed (2015) **showed that** running distinct simulations with independent populations on each parallel node decreased the communication cost. However, the periodic migration events, in which the most optimum populations from parallel nodes were exchanged, led to producing the highest-quality results. Thus, not only the cooperation phase should be preserved in parallel population-based optimization algorithms, but also it is beneficial to provide it for parallel single-solution optimization algorithms. The reason is that single-solution optimization algorithms are not equipped with strategies of avoidance of local optimum by appropriate cluster-wide node coordination and broader exploration (Majeed and Rao, 2017). Hence, knowledge-sharing becomes vital for both parallel single-solution and population-based optimization algorithms for adaptive and purposeful parallel exploitation and exploration of parameter search space.

Knowledge-sharing comes with two conditions: asynchronicity and low communication overhead. Zamani et al. (2021) parallelized Dynamically the Dimensioned Search (DDS) algorithm for parallel calibration of the SWAT model. They provided knowledge sharing to some degree for the algorithm but in a synchronous architecture. Their results showed negative impacts of knowledge sharing on parallel efficiency in parallel calibration of hydrologic models. This degradation of parallel efficiency, defined as the speedup gain relative to the scale of the parallel system (Asgari et al., 2022), was due to different configurations of computing units, such as various disk access times for reading/writing databases and potential failures throughout simulations on each node in a synchronous architecture. Besides, communication overhead in knowledge-sharing parallel systems should be minimized to ensure that adding the knowledge-sharing feature does not impact the overall parallel efficiency of these systems. Considering these two conditions, OpenMP and MPI parallelization paradigms are evaluated further.

OpenMP adopts Fork/Join (FJ) parallelism framework for shared-memory parallelization. In the FJ framework, task decomposition, parallel computation, and result combination are carried out repeatedly, which is suitable for applications that can be broken down recursively into several subproblems (Peng et al., 2017). This framework is more appropriate for fine-grained parallelization to avoid long latencies (the time-elapse between requesting a memory space and receiving a response (Silc et al., 1998)) in the result combination step derived from unequal processing speeds of parallel computing units. Synchronization in the FJ framework exists in the result combination, where all computing units must finish processing their assigned subproblems before proceeding with further unprocessed ones. By implementing fine-grained parallelization, which focuses on decomposing sequential problems to the maximum number of small and fast-to-finish parallelizable components, latencies in the result combination of FJ can be minimized (Corbellini et al., 2018; Lea, 2000). Another limitation of the OpenMP paradigm lies in its incapability of supplying horizontal scaling. Horizontal scaling is the capability of increasing involved computing units in the execution of the parallel application for more complex problems. In OpenMP-based systems, the number of computing units and the size of the shared memory bound the horizontal scaling (Kirk and Wen-Mei, 2016). In fact, long latencies in accessing the shared memory in these systems degrade theoretical parallel performance and consequently limit scalability.

The limitation of OpenMP-based systems paved the way for using the MPI parallelization technique to develop distributed-memory parallel systems for calibrating hydrologic models (Bomhof et al., 2019). In MPI-based systems, data transfer among parallel tasks are explicitly managed by sending/receiving messages. These explicit send-and-receive messages, which are I/O requests, ultimately lead to saturated I/O activities, long latencies due to multiple disk accesses, and parallel computing performance degradation (Ou et al., 2006). Another associated problem with the MPI parallelization technique is its low parallel productivity (Niu et al., 2021), which calls for more studies on parallel systems that can be easy-to-implement. Parallel productivity defines as the ability to develop High-Performance Computing (HPC) systems that are easier to program, which reduces software development cost and time; however, this ease should not decrease the overall system's parallel performance (Tolson and Shoemaker, 2007). Studies have referred to the low parallel productivity of MPI-based systems by mentioning the tight coupling between hydrologic models and these systems, which requires model reconstruction (Lin and Zhang, 2021; Zhang et al., 2021). Her et al. (2015) pointed out that significant modifications of sequential optimization algorithms and/or hydrologic model source codes were expected in developing MPI-based parallel calibration systems. Niu et al. (2021) encouraged hydrologists to adopt parallel computing in calibration of hydrologic models, which led to increased use of distributed hydrologic models and their enhancement. However, implementation of parallel systems should be relatively easy, with minimum requirements for pre-installation frameworks/libraries/packages.

The challenges in MPI-based systems along with their limitations in handling parallel task/node failures motivated researchers to implement Hadoop-based systems (Lin and Zhang, 2021; Zhang et al., 2021; Ma et al., 2022). However, several studies reported that the shared-memory of Hadoop-based systems and the I/O operations for node communications along with simultaneous disk access in computing units resulted in high I/O overheads and long latencies in these systems (Zamani et al., 2021; Ma et al., 2022; Zhang et al., 2021, Zhang et al., 2016). Lin and Zhang (2021) and Ma et al. (2021) pointed out that the Spark computing framework can reduce disk I/O overhead in Hadoop. These studies showed that increasing parallel tasks decreased the parallel efficiency in transferring big data blocks due to the distribution and operation of tasks in Spark (Ma et al., 2021). Similarly, due to Spark task management and data transfer, Spark might not be suitable for calibrating lightweight hydrological models with intensive communications (Lin and Zhang, 2021). Therefore, there is a research gap in developing fault-tolerant, portable, and easy-to-implement knowledge-sharing-based parallel calibration algorithms that require minimum communication overhead.

A new generation of parallel computing models called Partitioned Global Address Space (PGAS) has emerged to improve parallel productivity and performance. Unlike OpenMP-based systems, where only global memory is available, and MPI-based systems, where parallel computing units access *merely* local memory, each computing unit in the PGAS model has its local memory while part of it is *shared* with other computing units (global memory). In the PGAS model, computing nodes can read/write not only their local memories but also the remote/shared memory of each computing node, synchronously or asynchronously (De Wael et al., 2015; Dinan et al., 2010). Pang et al. (2022) showed that the asynchronous hydrologic model calibration has a 40%~70% improvement in computational time compared to the synchronous version. Parallel calibration of hydrologic models has an asynchronous nature due to different configurations of computing units, such as various disk access times for reading/writing databases and potential failures throughout simulations on each node. Hence, in nodal communications, only one side is aware of the communication event, which is called one-sided communication. This communication is a type of network communication in which one computing unit is responsible for identifying the sender/receiver of the data/knowledge and the size of data/knowledge. The other computing unit in network communication is only either the sender or receiver of data/knowledge. This communication is indispensable for asynchronous communication (Nishtala et al., 2011). Since the MPI parallelization paradigm requires matching send and receive messages to allow data transfer, which is called two-sided communication, it is inconvenient for knowledge-sharing parallel calibration of hydrologic models. On the other hand, PGAS languages are attractive for the parallel calibration of hydrologic models as they support one-sided communication. Chapel is a parallel programming language based on the PGAS model, which supports the implementation of productive and general-purpose HPC systems through its high-level abstractions for data and task parallelism. Gmys et al. (2020) studied high-performance programming languages (Chapel, Julia, and Python) for developing parallel metaheuristic optimization algorithms by considering C/OpenMP as the reference for their study. They concluded that Chapel was the only programming language with a higher level of parallel productivity than C/OpenMP in programming the metaheuristic optimization algorithms. The authors showed that Chapel had the smallest gap between serial and parallel relative performance, which proves its high scalability. Besides, Julia and Python do not have mature multi-threading support for programming parallel metaheuristics as Chapel does.

In this study, we proposed a knowledge-sharing parallel calibration approach as a general parallel framework for calibrating distributed hydrologic models using the Chapel programming language. Our parallelization approach stands out from all previously designed/implemented approaches/frameworks due to its unique features: 1) parallel Dynamically Dimensioned Search (DDS) algorithm with different-size perturbation and divided perturbation zones, 2) advanced database management, 3) asynchronous multithreading feature for fast and reliable knowledge-sharing, and 4) parallel node failure handling capability by addressing chapel's Q-threads tasking layer challenge for running hydrologic models. We implemented our proposed parallel calibration approach to calibrate the Integrated Modelling for the Watershed Evaluation of BMPs (IMWEBs) fully distributed hydrologic model (Liu et al., 2018) for the Catfish Creek watershed of Ontario in Canada. We evaluated the capability of our proposed approach in providing reliable model calibration results (model performance) along with parallel speed up and parallel efficiency. Our proposed approach achieved super-linear speedup and parallel efficiency above 75%. In addition, we proved that our approach has a low communication overheard along with the positive impact of knowledge-sharing in the convergence behaviour of the parallel DDS algorithm.

References:

Ab Wahab, M. N., S. Nefti-Meziani and A. Atyabi (2015). "A comprehensive review of swarm optimization algorithms." PloS one 10(5): e0122827.

Beheshti, Z. and S. M. H. Shamsuddin (2013). "A review of population-based meta-heuristic algorithms." Int. J. Adv. Soft Comput. Appl 5(1): 1-35.

Bomhof, J., Tolson, B. A., & Kouwen, N. (2019). Comparing single and multi-objective hydrologic model calibration considering reservoir inflow and streamflow observations. Canadian Water Resources Journal/Revue canadienne des ressources hydriques, 44(4), 319-336.

Budamala, V. and A. Baburao Mahindrakar (2021). "Enhance the prediction of complex hydrological models by pseudo-simulators." Geocarto International 36(9): 1027-1043.

Corbellini, A., D. Godoy, C. Mateos, S. Schiaffino and A. Zunino (2018). "DPM: A novel distributed large-scale social graph processing framework for link prediction algorithms." Future Generation Computer Systems 78: 474-480.

De Wael, M., S. Marr, B. De Fraine, T. Van Cutsem and W. De Meuter (2015). "Partitioned global address space languages." ACM Computing Surveys (CSUR) 47(4): 1-27.

Dinan, J., P. Balaji, E. Lusk, P. Sadayappan and R. Thakur (2010). Hybrid parallel programming with MPI and unified parallel C. Proceedings of the 7th ACM international conference on Computing frontiers.

Gmys, J., T. Carneiro, N. Melab, E.-G. Talbi and D. Tuyttens (2020). "A comparative study of high-productivity high-performance programming languages for parallel metaheuristics." Swarm and Evolutionary Computation 57: 100720.

Hadka, D., & Reed, P. (2015). Large-scale parallelization of the Borg multiobjective evolutionary algorithm to enhance the management of complex environmental systems. Environmental modelling & software, 69, 353-369.

Her, Y., R. Cibin and I. Chaubey (2015). "Application of parallel computing methods for improving efficiency of optimization in hydrologic and water quality modeling." Applied Engineering in Agriculture 31(3): 455-468.

Huo, J., L. Liu and Y. Zhang (2018). "An improved multi-cores parallel artificial Bee colony optimization algorithm for parameters calibration of hydrological model." Future Generation Computer Systems 81: 492-504.

Kan, G., X. He, L. Ding, J. Li, Y. Hong, D. Zuo, M. Ren, T. Lei and K. Liang (2018). "Fast hydrological model calibration based on the heterogeneous parallel computing accelerated shuffled complex evolution method." Engineering Optimization 50(1): 106-119.

Kan, G., He, X., Ding, L., Li, J., Hong, Y., & Liang, K. (2020). Heterogeneous parallel computing accelerated generalized likelihood uncertainty estimation (GLUE) method for fast hydrological model uncertainty analysis purpose. Engineering with Computers, 36(1), 75-96.

Kan, G., Lei, T., Liang, K., Li, J., Ding, L., He, X., . . . Bao, Z. (2016). A multi-core CPU and many-core GPU based fast parallel shuffled complex evolution global optimization approach. Ieee transactions on parallel and distributed systems, 28(2), 332-344.

Khakbaz, B., B. Imam, K. Hsu and S. Sorooshian (2012). "From lumped to distributed via semi-distributed: Calibration strategies for semi-distributed hydrologic models." Journal of Hydrology 418: 61-77.

Kim, J., & Ryu, J. H. (2019). Quantifying the performances of the semi-distributed hydrologic model in parallel computing—A case study. Water, 11(4), 823.

Kirk, D. B. and W. H. Wen-Mei (2016). Programming massively parallel processors: a hands-on approach, Morgan kaufmann.

Lea, D. (2000). A java fork/join framework. Proceedings of the ACM 2000 conference on Java Grande.

Li, Q., Chen, X., Luo, Y., Lu, Z., & Wang, Y. (2015). A new parallel framework of distributed SWAT calibration. Journal of Arid Land, 7(1), 122-131.

Lin, Q. and D. Zhang (2021). "A scalable distributed parallel simulation tool for the SWAT model." Environmental Modelling & Software 144: 105133.

Liu, Y., W. Yang, H. Shao, Z. Yu and J. Lindsay (2018). "Development of an integrated modelling system for evaluating water quantity and quality effects of individual wetlands in an agricultural watershed." Water 10(6): 774.

Ma, J., K. Rao, R. Li, Y. Yang, W. Li and H. Zheng (2022). "Improved Hadoop-based cloud for complex model simulation optimization: calibration of SWAT as an example." Environmental Modelling & Software 149: 105330.

Ma, Y., P.-a. Zhong, B. Xu, F. Zhu, Q. Lu and H. Wang (2021). "Spark-based parallel dynamic programming and particle swarm optimization via cloud computing for a large-scale reservoir system." Journal of Hydrology 598: 126444.

Majeed, M. M. and P. S. Rao (2017). Optimization of CMOS analog circuits using grey wolf optimization algorithm. 2017 14th IEEE India Council International Conference (INDICON), IEEE.

Nishtala, R., Y. Zheng, P. H. Hargrove and K. A. Yelick (2011). "Tuning collective communication for Partitioned Global Address Space programming models." Parallel Computing 37(9): 576-591.

Niu, W.-j., Z.-k. Feng, B.-f. Feng, Y.-s. Xu and Y.-w. Min (2021). "Parallel computing and swarm intelligence based artificial intelligence model for multi-step-ahead hydrological time series prediction." Sustainable Cities and Society 66: 102686.

Ou, L., X. Chen, X. B. He, C. Engelmann and S. L. Scott (2006). Achieving computational I/O efficiency in a high performance cluster using multicore processors. Proceedings of the, Citeseer.

Pang, M., Shoemaker, C. A., & Bindel, D. (2022). Early termination strategies with asynchronous parallel optimization in application to automatic calibration of groundwater PDE models. Environmental modelling & software, 147, 105237.

Peng, Y., A. Peng, X. Zhang, H. Zhou, L. Zhang, W. Wang and Z. Zhang (2017). "Multi-Core parallel particle swarm optimization for the operation of Inter-Basin water transfer-supply systems." Water Resources Management 31(1): 27-41.

Prügel-Bennett, A. (2010). "Benefits of a population: Five mechanisms that advantage population-based algorithms." IEEE Transactions on Evolutionary Computation 14(4): 500-517.

Quinn, M. J. (1994). Parallel computing theory and practice, McGraw-Hill, Inc.

Razavi, S., B. A. Tolson, L. S. Matott, N. R. Thomson, A. MacLean and F. R. Seglenieks (2010). "Reducing the computational cost of automatic calibration through model preemption." Water Resources Research 46(11).

Refsgaard, J. C. (1997). "Parameterisation, calibration and validation of distributed hydrological models." Journal of hydrology 198(1-4): 69-97.

Silc, J., Robic, B., & Ungerer, T. (1998). Asynchrony in parallel computing: From dataflow to multithreading. Parallel and Distributed Computing Practices, 1(1), 3-30.

Singh, V. P. and D. K. Frevert (2003). Watershed modeling. World Water & Environmental Resources Congress 2003.

Tolson, B. A. and C. A. Shoemaker (2007). "Dynamically dimensioned search algorithm for computationally efficient watershed model calibration." Water Resources Research 43(1).

Vrugt, J. A., Nualláin, B. O., Robinson, B. A., Bouten, W., Dekker, S. C., & Sloot, P. M. (2006). Application of parallel computing to stochastic parameter estimation in environmental models. Computers & Geosciences, 32(8), 1139-1155.

Xia, W., Shoemaker, C., Akhtar, T., & Nguyen, M.-T. (2021). Efficient parallel surrogate optimization algorithm and framework with application to parameter calibration of computationally expensive three-dimensional hydrodynamic lake PDE models. Environmental modelling & software, 135, 104910.

Xu, D.-m., W.-c. Wang, K.-w. Chau, C.-t. Cheng and S.-y. Chen (2013). "Comparison of three global optimization algorithms for calibration of the Xinanjiang model parameters." Journal of Hydroinformatics 15(1): 174-193.

Zamani, M., N. K. Shrestha, T. Akhtar, T. Boston and P. Daggupati (2021). "Advancing model calibration and uncertainty analysis of SWAT models using cloud computing infrastructure: LCC-SWAT." Journal of Hydroinformatics 23(1): 1-15.

Zhang, D., Chen, X., Yao, H., & James, A. (2016). Moving SWAT model calibration and uncertainty analysis to an enterprise Hadoop-based cloud. Environmental modelling & software, 84, 140-148.

Zhang, A., T. Li, Y. Si, R. Liu, H. Shi, X. Li, J. Li and X. Wu (2016). "Double-layer parallelization for hydrological model calibration on HPC systems." Journal of Hydrology 535: 737-747.

Zhang, D., B. Lin, J. Wu and Q. Lin (2021). "GP-SWAT (v1. 0): a two-level graph-based parallel simulation tool for the SWAT model." Geoscientific Model Development 14(10): 5915-5925.

Zhang, X., R. Srinivasan and M. Van Liew (2009). "Approximating SWAT model using artificial neural network and support vector machine 1." JAWRA Journal of the American Water Resources Association 45(2): 460-474.