modelling

MDPI

Review

Heterogeneous Compute Clusters and Massive Environmental Simulations Based on the EPIC Model

Nikolay Khabarov [1,*], Alexey Smirnov [1,2], Juraj Balkovič [1,3], Rastislav Skalský [1,4], Christian Folberth [1], Marijn Van Der Velde [5] and Michael Obersteiner [1,6]

1 Ecosystems Services and Management Program (ESM), International Institute for Applied Systems Analysis (IIASA), A-2361 Laxenburg, Austria; smirnov@iiasa.ac.at (A.S.); balkovic@iiasa.ac.at (J.B.); skalsky@iiasa.ac.at (R.S.); folberth@iiasa.ac.at (C.F.); oberstei@iiasa.ac.at (M.O.)
2 Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, 119991 Moscow, Russia
3 Department of Soil Science, Faculty of Natural Sciences, Comenius University in Bratislava, 842 15 Bratislava, Slovakia
4 National Agricultural and Food Centre, Soil Science and Conservation Research Institute, 824 80 Bratislava, Slovakia
5 European Commission, Joint Research Centre, 21027 Ispra, Italy; marijn.van-der-velde@ec.europa.eu
6 Environmental Change Institute, Oxford University Centre for the Environment, Oxford OX1 3QY, UK
* Correspondence: khabarov@iiasa.ac.at

Received: 4 November 2020; Accepted: 30 November 2020; Published: 4 December 2020

Abstract: In recent years, the crop growth modeling community invested immense effort into high resolution global simulations estimating inter alia the impacts of projected climate change. The demand for computing resources in this context is high and expressed in processor core-years per one global simulation, implying several crops, management systems, and a several decades time span for a single climatic scenario. The anticipated need to model a richer set of alternative management options and crop varieties would increase the processing capacity requirements even more, raising the looming issue of computational efficiency. While several publications report on the successful application of the original field-scale crop growth model EPIC (Environmental Policy Integrated Climate) for running on modern supercomputers, the related performance improvement issues and, especially, associated trade-offs have only received, so far, limited coverage. This paper provides a comprehensive view on the principles of the EPIC setup for parallel computations and, for the first time, on those specific to heterogeneous compute clusters that are comprised of desktop computers utilizing their idle time to carry out massive computations. The suggested modification of the core EPIC model allows for a dramatic performance increase (order of magnitude) on a compute cluster that is powered by the open-source high-throughput computing software framework HTCondor.

Keywords: EPIC model; high performance computing (HPC); high-throughput computing (HTC); HTCondor; massive spatio-temporal modeling; legacy source code; environmental simulation; heterogeneous compute clusters; crop model; agriculture; climate change

1. Introduction

The global high resolution simulations that were carried out recently by the crop growth modeling community (GGCMI) [1] required a considerable amount of computing power expressed in processor core-years, as such computations normally include model runs for several years, crops, management systems, and climatic scenarios. One of the questions particularly addressed by these large scale modeling experiments was the inter-comparison of different agricultural models. These efforts were part of a larger research also looking at sectors other than agriculture [2] and provided the inputs to

economic modeling [3]. The demand for computing power in similar applications will likely increase in the future as new input data at a finer spatio-temporal resolution become available and a richer set of crops and management systems will be demanded.

There is a vast amount of knowledge accumulated in software products–environmental models–that have developed over long periods of time. While the computer technologies evolve, the old code becomes less suitable for new computing environments as compared to code that is specifically developed for modern infrastructure. A full re-implementation of older software from scratch to better fit new operating conditions seems to be challenging due to a range of reasons, including a lack of documentation, specifics of compilers used, and mostly due to an immense effort needed. From this perspective, it seems to be reasonable instead of a full re-implementation to adapt the existing source code and the new operating environment to better fit together. This adaptation requires extensive testing in order to identify the computational bottlenecks and interventions needed.

The Environmental Policy Integrated Climate model (EPIC) [4] is a crop growth model that has been widely used by the global gridded crop modeling community. In earlier literature describing the EPIC cluster applications, the aspects of Linux cluster setup [5] and inclusion of the MPI technology [6] were covered in detail. Other papers went further in the simulation performance optimization (even though for a model other than EPIC) and considered the full re-implementation of the model in different programming languages [7]. In this paper, we focus on increasing the EPIC model performance on compute clusters, document the decisions that we have made, and point to relevant trade-offs between the effort invested and the efficiency gains. Special emphasis is put onto aspects of such adaptations that are relevant to heterogeneous clusters—those that can be formed by utilizing the idle time of normal desktop computers and, hence, provide a low cost (and for medium-scale applications also low environmental footprint) cluster-based computing. However, as compared to dedicated clusters, the compute nodes of such systems have fluctuating availability, uneven performance, and lack uniform configuration. For the purposes of managing the compute cluster, we employ HTCondor—an open-source cross-platform high-throughput computing software framework for coarse-grained distributed parallelization of computationally intensive tasks [8].

2. Design and Implementation

2.1. General EPIC Setup

EPIC is a process-based cropping systems model that was developed in order to simulate crop growth and yield at a field scale, as driven by site environmental conditions and crop management [4]. The gridded EPIC-IIASA infrastructure [9] uses EPIC v.0810 in order to estimate crop yields at a global scale, with 5 arc-min. spatial resolution. EPIC-IIASA runs the EPIC model for more than 120,000 spatial simulation units (SimUs) that are derived from intersecting soil and topography units, administrative borders and climate grids following a set of criteria for internal homogeneity of SimU. Each SimU is represented by one (default) or more homogeneous fields with "representative" soil, topography, and present weather. Globally available data sources on climate, soils, and land use are used to construct the simulation units [10]. A large set of crop management scenarios (crop varieties, fertilization and irrigation options, and soil conservation practices) and climate change projections simulated for each SimU are a common workload for the EPIC-IIASA model [11].

2.2. Input/Output Data Formats

The EPIC model is very data intensive and the number of variables and parameters describing initial state of a site (field), its functioning (bio-physical processes), and present management is approximately 300 [12]. In addition to that static initial input data, dynamic inputs are required and daily weather is the largest dataset among those that are used by EPIC. All of the input files for EPIC are in the text format with the values being located at the fixed positions within a file. In terms of input/output efficiency, the plain text format is inferior to modern binary formats as, e.g., netCDF [13],

because each number that is expressed as a plain text has to be translated to a floating point value (or a floating point value has to be converted to text), even though efficient libraries exist [14]. Despite that inefficiency, the effort–benefit trade-off speaks to leaving the format as plain text for two reasons. First, for the main part of the (static) variables, the overhead implied with this conversion is negligible when compared to the resources that are needed to carry out actual computation; moreover, this is true, even for the daily climate data, as translating daily values from text to floating point only comprises a small fraction of the whole associated computation for a day. Second, while a considerable effort would be needed to modify the source code of the model to consistently use binary input data, it would still not allow for straight-forward use of available weather datasets (see Section 2.3 below).

Similar consideration and justification is also valid for EPIC outputs and plain text works acceptably well, unless daily outputs are required. That would create an immense amount of output data, rendering plain text format unusable; however, in practical global applications, this finer temporal granularity is not used, as monthly or annual outputs are deemed to be sufficient at the time of writing this note.

2.3. Weather Data Pre-Processing

Before a weather dataset (that is usually available in a binary netCDF format) can be used to run EPIC, it needs to be converted to the plain text format. This procedure is more sophisticated than just taking floating point numbers one-by-one from the binary file and writing each as a text. The challenge here is that one run of EPIC needs the whole record day-by-day for a particular grid cell, whereas, in the commonly used weather datasets e.g., AgMERRA [15,16], ISI-MIP [17,18], Princeton [19,20], for each day there is a whole map in the file instead of just one value, and filtering out one grid cell is inefficient on a hard disk based storage devices (and also newer solid state drives), because, in that case, the whole map has to be read or skipped (employing a time expensive seek operation) repeated as many times as there are days in the dataset multiplied by the number of grid cells in it. Accordingly, more sophisticated procedures need to be used to carry out that "transposition" of the time dimension in such a dataset. In a practical realization of such a procedure, one has to make a decision based on the trade-off between the number of seek operations and available RAM (random access memory) in a processing machine, optionally creating intermediate fragments of time-transposed subsets of original data and merging them at a final step. These considerations are also valid for the final post-processing stage, where data pertinent to individual SimUs are aggregated to e.g., annual maps. However, here the conversion has to be carried out in an opposite direction: from individual SimUs time series to a series of maps.

2.4. Parallel Runs on a Cluster

There are several aspects of general consideration when moving an application developed for running on a single-computer to a cluster, where many instances of such an application will be running in parallel, even in case these runs are isolated from each other by using resources of only a local node i.e., local disk storage. Some examples of those are highlighted below without the goal to be exhaustive, but relevant to the EPIC model.

Non-interactive execution is the very nature of operating a cluster node. The jobs are submitted to a cluster in a batch [21] and, after the processing, the outputs and/or error reports are delivered to the submitting user. Unlike in the case of a single computer run, it is not possible to work with each instance interactively and answer questions regarding whether to continue program execution or not should any non-standard input be detected by the model. The possibility for a user to interact with the program creates much needed flexibility when running it on a single machine, yet it creates a problem when running it on a cluster node, where a typical outcome would be canceling the job by a timeout when the program waits for a user confirmation whereas it is not going to come. Such interactive requests from a program have to be eliminated e.g., by reporting an error to a log file and carrying on with the next grid cell before the code can be run on a cluster; otherwise, the need to wait

until a timeout occurs would delay the simulation outputs and, by that, consume valuable time of the research staff and lower the use efficiency of the compute resources within an organization.

Tracking of model crashes is another issue that needs attention within a cluster environment. A wrapper or launcher might help here in logging errors, keeping track of problematic cells (or generally subsets of problematic input combinations e.g., weather plus management practices), and restarting simulations for the next portion of data.

In case if a network storage is used to feed model runs on cluster nodes, the issue of input/output optimization may turn out to be important. This can play a considerable role in the overall model performance if the number of files opened per run, number of seek operations is large and/or there are shared files. While local disk win shared network storage because they provide the intended environment for a model run, this mode of operation implies data duplication and transfer time overheads that negatively impact model performance and create an excessive load on the network, especially if the network is also used for other purposes than just cluster runs, which is naturally the case in heterogeneous clusters that can be managed by HTCondor.

An effect that is related to the network storage performance and excessive read/rewind operations can be seen in practice when running a subset of EPIC simulations as one serial batch. It turns out that there is a major speed limitation, because, for each iteration, EPIC reads through the large index files (e.g., OPSCCOM and SITECOM), listing file names that are specific to a particular virtual agricultural field—SimU. Accordingly, a considerable speed gain can be achieved by not running EPIC in its native batch mode, but linking the relevant input files for each SimU to placeholder files with a constant name prior to invoking EPIC. The resulting speed increase observed was about the factor of 2 to 3.

A note of caution should be given also with regard to compilers. While there exist standards for programming languages (e.g., Fortran in EPIC case), there are still subtle points in compilers configuration to make source code work as intended with an alternative compiler. As the authors found out, the EPIC outputs differ when compiled by GNU Fortran from those when Intel Fortran is used (officially supported by the EPIC developers). One of the suspects causing this effect is the floating point comparison operation working differently in those two compilers.

3. EPIC-HTCondor Case Study and Benchmarking

A typical set of global runs as in e.g., ISI-MIP project [18] implies a processing of terabytes of data and millions of CPU-hours i.e., weeks of computation on a thousand-core cluster. Despite an apparent need to employ substantial computing power that is routinely available on dedicated remote clusters, the data size may turn out to be quite an obstacle due to the bandwidth limits and need to transfer large data between remote systems. If data are transferred to an in-house processing facility, it requires the time consuming data transfer procedure to be carried out only one time, whereas transferring data from the original source to a dedicated remote cluster and copying the results back from that remote cluster to an in-house storage system (for backup and further processing) requires the time consuming data transfer procedure to be completed twice (for two different datasets of comparable size—model's input and output). There are more limiting aspects to using dedicated clusters in practice, as e.g., large number of small files operated by EPIC, which is generally not the mode of operation of modern highly parallel network storage that is usually oriented towards high-speed throughput of large chunks of data, and not accessing a huge number of rather small data pieces (files). The high cost of ownership of a dedicated cluster and the potential benefits of utilizing a heterogeneous in-house cluster have apparently supported the success and popularity of the HTCondor cluster software [22]. While the authors have experience with other cluster architectures, the case study that is presented here is focused, to a large degree, on a HTCondor-based cluster.

3.1. Custom Load Balancer

The main task of the custom load balancer (CLB), which the authors implemented for EPIC-IIASA, is to take into account various speeds of cluster nodes and distribute calculation sub-tasks in small

chunks to the nodes as they complete the chunks received earlier. Keeping individual SimUs to be modeled "behind" a CLB (and not exposed to the HTCondor's job manager) helps to maintain the cluster's job queue in an overseeable state.

A useful feature of the EPIC model is its ability to carry out calculation in batches, so that several simulation units (SimU, "representative" fields) can be processed by starting the EPIC model only once. That allows for a faster execution (unless the index files become too large as discussed in the Section 2.4) as compared to the one-by-one alternative starting EPIC separately for each SimU, because the number of SimUs is large and loading the model into RAM is far from instantaneous even taking advantage of a cached disk access. The implementation of that "batched": approach allows for boosting the efficiency; however, error handling within this context needs an additional effort. If one simulation unit (SimU) within a batch fails (e.g., for a reason of incorrect input data), then it leads to the necessity of re-running all SimUs from that batch individually in order to report problematic SimUs and allow for a detailed inspection. Overall, as these "problematic" SimUs are rather exceptional and their share is usually below 1% in EPIC-IIASA. The logic for tracking and reporting problematic SimUs is included in the custom load balancer that was implemented by the authors.

Because of the very nature of HTCondor to run jobs on compute nodes having limited availability, it does automatically restart jobs that were interrupted as the computer they were running on was claimed by a user. These jobs are restarted from scratch on another available machine under the same job identifier (JID). The use of JIDs along with a custom sub-task tracking implemented within CLB allows for an efficient resumption of a computation batch by only calculating the leftover part of the original batch. This approach is implemented within the CLB, which technically is an EPIC wrapper—an envelope setting up an individual EPIC run on a cluster node and starting the EPIC executable file while linking to a database in order to keep track of the whole calculation at the SimU level. CLB follows a semi-decentralized approach to the load balancing implementation, where the entire logic is encapsulated within the same wrapper running simultaneously on multiple nodes and accessing the same central database. The absence of a central application level manager does greatly simplify the system's setup. However, initially, locking the central database table containing information on scheduled simulations turned out to be a bottleneck badly affecting the overall cluster performance. The issue was resolved by splitting the whole SimU table into a cluster of smaller tables that are pertinent to individual cluster jobs that can be locked independently without affecting each other. The time overhead created by this implementation is estimated to be about 1% of the total computation time. While allowing for high performance, this approach provides an efficient way for the tracking of computed SimUs via a summary table available to a user via the standard client tools accessing the SQL database.

In addition to the main CLB logic that is described above, the EPIC wrapper also performs rather simple technical tasks of configuring a cluster node for a model run e.g., mounting necessary network shares, including those that serve as a temporary space for storing intermediate model outputs.

3.2. Input/Output and Shared Network Storage

While the computation speed on a single computer is mostly limited by the CPU power and depends on the processor architecture, operating frequency, and the number of cores for parallel runs, there is another factor of large importance in practical cluster applications—the storage sub-system performance. From the authors' experience, this is a multifaceted issue being impacted by multiple factors, such as: the total number of files involved in the computation, the share of those commonly used per one SimU (e.g., several SimUs sharing the same weather data), the sizes of individual files, the number, availability, and performance of individual nodes participating in a parallel run, the type of the storage system, and the network configuration. Here, for comparison purposes, we present the results of the impact of the number of nodes and different storage hardware. The motivation for the exploration in this direction was provided by the fact that, in the process of an upgrade of the network storage system from an older VNX5400 (http://vnx5400.com/) to a newer ZS5 (http:

//www.oracle.com/us/products/servers-storage/storage/nas/ds-nas-storage-zs5-3158594.pdf) used for the EPIC cluster runs, the authors faced an unexpected performance downgrade. The original EPIC binary code turned out to be three to four times slower on the newer storage as reported in Table 1, columns "Original" and similar to the CPU-optimized code "Opt.Calc".

Table 1. The performance of a 100-instances Environmental Policy Integrated Climate model (EPIC) job on HTCondor cluster: the original binary code "Original", a source code compiled with optimization for calculation speed "Opt.Calc", and with added read–only mode for all input files "Opt.Calc + RO". Units are SimU/minute.

Storage	Original	Opt.Calc	Opt.Calc + RO
VNX5400	717	1146	1444
ZS5	241	298	1553

After the examination of the model's source code and using the process monitoring tool Process Explorer that was developed for the Microsoft Windows operating system by Mark Russinovich (https://docs.microsoft.com/en-us/sysinternals/downloads/process-explorer), the problem with locking input files for exclusive access was revealed. The files in question are supposed to be read-only, as these are inputs to the model that are not supposed to be modified in the course of the model run. The default file access mode used in EPIC file operations turned out to be read and write because of using the plain Fortran `OPEN` command. On machines running Microsoft Windows operating system this has lead to runtime failures when multiple clients requested exclusive write mode on a shared file storage. (Machines running Linux did not expose this behaviour, because the `OPEN` command did not implicitly put an exclusive lock on a file.) The Intel Fortran runtime handled the problem on the fly and provided a remedy by changing the file open mode to read only, which was, in the end, sufficient for a successful model run, yet it led to considerable delays in input–output operations strongly downgrading the performance.

While the cause of the problem was not trivial to localize, the solution to it was pretty obvious: the plain Fortran `OPEN` statements were amended with an explicit `ACTION='READ'` specifier for input files, preserving the intended write mode for output files. The modification affected only one `OPENV` subroutine of EPIC. The performance of the optimized model "Opt.Calc + RO" for a different number of simultaneously working cluster nodes on different storage systems is presented in Table 2.

Table 2. Performance of the optimized model "Opt.Calc + RO" on two different storage systems and for two different numbers of simultaneously working cluster nodes. Units are SimU/minute.

Storage/Nodes Count	100	300
VNX5400	1444	2607
ZS5	1553	4483

While the performance of the older storage VNX5400 is pretty much comparable with that of the newer storage ZS5 for a 100-nodes run (Table 2, column "100"), it suffers more under a heavy load delivering considerably lower performance (Table 2, column "300"), and this is what is expected, unlike the opposite performance of the original code (Table 1, the "Original" column).

A closer look into the model performance, depending on the number of "active" cluster nodes, confirms the expectation that the network storage performance scales up non-linearly, demonstrating a degradation for a heavier load coming from the cluster, as shown on Figure 1. There we present the results that were obtained for an older storage—NAS7310 (https://blogs.oracle.com/ahl/sun-storage-7310)—in order to illustrate a more vivid performance degradation. The newer storage ZS5 performance does scale linearly for the relatively small number of cores available at the cluster (Table 2).

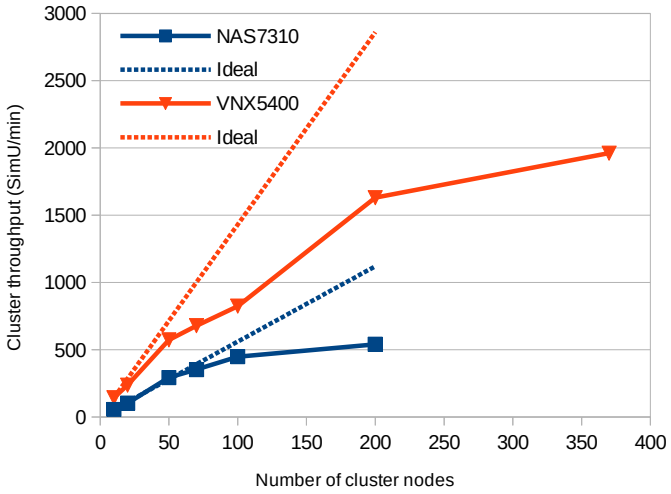

Figure 1. EPIC calculation speed for a test run on cluster depending on the number of nodes simultaneously running the model.

4. Discussion

This paper highlights some of the directions for the benchmarking and revision of legacy code. In practice, an efficient system setup is not obvious and extensive testing is required in order to obtain expected performance gains of an old code on a newer infrastructure. Synthetic tests usually deliver imprecise estimates, as the specifics of calculations carried out by a model, the number of files opened per running process, and read/write access patterns cannot be realistically reproduced; this is why only a benchmarking run of a real model can ultimately be used in order to estimate its performance under different conditions.

Heterogeneous cluster environments pose additional specific challenges. An implementation of an efficient custom load balancer can help in addressing some of those—it is advantageous on taking into account the difference in performance of individual cluster nodes. In addition to that, CLB's use allows for avoiding the need for creation of self-contained execution packages and respective data copying to and from the cluster nodes, as described in [5]. The proposed semi-decentralized approach to the CLB implementation only leaves a coordinating function to the load balancer, while the EPIC instances running on cluster's cores independently access the required data files. This allows for avoiding a potential bottleneck where a message passing interface (MPI) master routine [6] handles, in a centralized way, both model's massive inputs and outputs and additionally coordinates task assignments. A co-benefit of the proposed "lighter" CLB approach is its isolation from the core EPIC model eliminating the source code re-integration need upon new model release.

While the tracking of model crashes needs special attention when running a model in a non-interactive mode on any type of a compute cluster, a shared storage on dedicated clusters is commonly configured differently from what is accessible from an in-house desktop PC-based HTCondor environment. In practice, storage is the first suspect for a potential bottleneck.

Simple adjustments to the source code, such as a less demanding file access mode, can better fit the parallel processing environment and drastically improve performance. Because these adjustments are only possible if the model's source code is open, there is a great benefit in the open-source distribution paradigm for both prolonging the lifetime of a model and preserving the associated accumulated scientific knowledge contained therein.

5. Conclusions

While the development effort can be saved and a vast accumulated scientific knowledge can be used, the problem of the efficient use of a mature model, i.e., legacy source code on a new parallel computing infrastructure is challenging. This challenge is multidimensional, as it implies the need

to consider and analyse many interconnected components and factors impacting the performance of the modeling system. In this paper, we have discussed an extensive set of those factors and relevant trade-offs specific to the EPIC model highlighting the aspects that have not yet received enough attention in earlier publications. Those include input–output text/binary data formats and related pre-processing of the weather data being the largest input, shared network storage use optimization, and bandwidth limitations when transferring large datasets between remote systems being an argument for in-house computing.

Unlike similar papers, this manuscript does not report on any particular biophysical results estimated by EPIC; instead, it focuses, in great detail, on the technical side of the modeling system setup and, presents, for the first time, benchmarking results for a heterogeneous compute cluster managed by the open-source HTCondor framework. A custom load balancer following an alternative to existing implementations semi-decentralized approach is proposed and its high efficiency is illustrated for the number of compute cores available at the cluster. A minor modification of the EPIC model source code is suggested that provides a dramatic performance increase on HTCondor, which has proven itself as an efficient, low-cost, and low environmental footprint parallel computing solution.

The discussed ideas for source code and run-time environment optimization for compute clusters can be useful beyond the presented specific model application.

Author Contributions: N.K., R.S., J.B., and C.F. worked on pre- and post-processing of EPIC input and output data. J.B. and C.F. carried out EPIC simulations. A.S. worked on setting up cluster infrastructure for the model and developed the custom load balancer. N.K. and A.S. worked on the optimization of the model's code for the cluster infrastructure and together with C.F. carried out various performance tests on different cluster facilities. All co-authors discussed the preliminary results and their interpretation. N.K. wrote the manuscript with contributions from all co-authors. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the project "Delivering Incentives to End Deforestation: Global Ambition, Private/Public Finance and Zero-Deforestation Supply Chains" funded by the Norwegian Agency for Development Cooperation under agreement number QZA-0464, QZA-16/0218. This research was funded by the European Research Council Synergy Grant number 610028 Imbalance-P: Effects of phosphorus limitations on Life, Earth system and Society (Seventh Framework Programme of the European Union).

Acknowledgments: The authors would like to thank the developers of the EPIC model and particularly Jimmy Williams (Texas A&M AgriLife Research, USA) for his support at the early stage of this work. The authors greatly appreciate the help and support from the IIASA's Information and Communication Technologies (ICT) Department, namely Joe Undercoffer, Hans Mayer, Helmut Klarn, Norbert Eder, Serge Medow, and their colleagues. Nikolay Khabarov, Christian Folberth, and Alexey Smirnov would like to thank Jacob Schewe and Karsten Kramer from the Potsdam Institute for Climate Impact Research (PIK) for providing access to the PIK's HPC cluster. Nikolay Khabarov and Alexey Smirnov are thankful to Jussi Heikonen and Antti Pursula from the Center for Scientific Computing, Finland (https://research.csc.fi/), Herbert Störi and Jan Zabloudil from the Vienna Scientific Cluster, Austria (https://vsc.ac.at/) for providing access to the respective high performance compute clusters. The authors are thankful for the support they received from their colleagues Anni Reissell, Ligia Azevedo, and the long time collaborator Erwin Schmid from the University of Natural Resources and Life Sciences in Vienna, Austria. The authors appreciate the efforts of the HTCondor developers and users community. The authors are thankful to the three anonymous reviewers who have helped to improve the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

CLB	custom load balancer
CPU	central processing unit(s) of a computer
EPIC	Environmental Policy Integrated Climate model
EPIC-IIASA	global gridded crop model powered by EPIC v.0810
HPC	high performance computing
HTC	high throughput computing
JID	job identifier
IIASA	International Institute for Applied Systems Analysis, Laxenburg, Austria
RAM	random access memory

References

1. Global Gridded Crop Model Intercomparisons (GGCMI) Initiative. Available online: https://agmip.org/aggrid-ggcmi/ (accessed on 30 October 2020).
2. The Inter-Sectoral Impact Model Intercomparison Project. Available online: https://www.isimip.org/ (accessed on 30 October 2020).
3. Agricultural Model Intercomparison and Improvement Project. Available online: https://agmip.org/ (accessed on 30 October 2020).
4. Williams, J.R.; Singh, V. Computer models of watershed hydrology, chap. In *The EPIC Model*; Water Resources Publications: Highlands Ranch, CO, USA, 1995; pp. 909–1000.
5. Nichols, J.; Kang, S.; Post, W.; Wang, D.; Bandaru, V.; Manowitz, D.; Zhang, X.; Izaurralde, R. HPC-EPIC for high resolution simulations of environmental and sustainability assessment. *Comput. Electron. Agric.* **2011**, *79*, 112–115. [CrossRef]
6. Kang, S.; Wang, D.; Nichols, J.; Schuchart, J.; Kline, K.; Wei, Y.; Ricciuto, D.; Wullschleger, S.; Post, W.; Izaurralde, R. Development of mpi_EPIC model for global agroecosystem modeling. *Comput. Electron. Agric.* **2015**, *111*, 48–54. [CrossRef]
7. de Wit, A.; Boogaard, H.; Fumagalli, D.; Janssen, S.; Knapen, R.; van Kraalingen, D.; Supit, I.; van der Wijngaart, R.; van Diepen, K. 25 years of the WOFOST cropping systems model. *Agric. Syst.* **2019**, *168*, 154–167. [CrossRef]
8. Center for High Throughput Computing at UW-Madison. HTCondor Web Site. Available online: https://research.cs.wisc.edu/htcondor/ (accessed on 30 October 2020).
9. Folberth, C.; Khabarov, N.; Balkovič, J.; Skalský, R.; Visconti, P.; Ciais, P.; Janssens, I.; Peñuelas, J.; Obersteiner, M. The global cropland-sparing potential of high-yield farming. *Nat. Sustain.* **2020**, *3*, 281–289. [CrossRef]
10. Skalský, R.; Tarasovičová, Z.; Balkovič, J.; Schmid, E.; Fuchs, M.; Moltchanova, E.; Kindermann, G.; Scholtz, P. GEO-BENE Global Database for Bio-Physical Modeling v. 1.0 (Concepts, Methodologies and Data). 2009. Available online: https://geo-bene.project-archive.iiasa.ac.at/files/Deliverables/Geo-BeneGlbDb10(DataDescription).pdf (accessed on 30 October 2020).
11. Balkovič, J.; Skalský, R.; Folberth, C.; Khabarov, N.; Schmid, E.; Madaras, M.; Obersteiner, M.; van der Velde, M. Impacts and uncertainties of +2°C of climate change and soil degradation on European crop calorie supply. *Earth's Future* **2018**, *6*, 373–395. [CrossRef] [PubMed]
12. Gerik, T.; Williams, J.; Francis, L.; Greiner, J.; Magre, M.; Meinardus, A.; Steglich, E.; Taylor, R. *Environmental Policy Integrated Climate Model–User's Manual Version 0810*; Blackland Research and Extension Center, Texas A&M AgriLife: Temple, TX, USA, 2014.
13. UCAR Community Programs. Network Common Data form (NetCDF). 2020. Available online: https://www.unidata.ucar.edu/software/netcdf/ (accessed on 30 October 2020).
14. Galbreath, N. Fast C–String Transformations: Stringencoders Library in C Programming Language. 2007–2020. Available online: https://github.com/client9/stringencoders (accessed on 30 October 2020).
15. Ruane, A. AgMERRA and AgCFSR Climate Forcing Datasets for Agricultural Modeling. 2017. Available online: https://data.giss.nasa.gov/impacts/agmipcf/ (accessed on 30 October 2020).
16. Ruane, A.; Goldberg, R.; Chryssanthacopoulos, J. Climate forcing datasets for agricultural modeling: Merged products for gap-filling and historical climate series estimation. *Agric. For. Meteorol.* **2015**, *200*, 233–248. [CrossRef]
17. Inter-Sectoral Impact Model Intercomparison Project–Data Archive. Available online: https://esg.pik-potsdam.de/projects/isimip/ (accessed on 30 October 2020).
18. Warszawski, L.; Frieler, K.; Huber, V.; Piontek, F.; Serdeczny, O.; Schewe, J. The inter-sectoral impact model intercomparison project (ISI–MIP): Project framework. *Proc. Natl. Acad. Sci. USA* **2014**, *111*, 3228–3232. [CrossRef] [PubMed]
19. Global Meteorological Forcing Dataset for Land Surface Modeling. 2014. Available online: http://hydrology.princeton.edu/data.pgf.php (accessed on 30 October 2020).
20. Sheffield, J.; Goteti, G.; Wood, E. Development of a 50-Year High-Resolution Global Dataset of Meteorological Forcings for Land Surface Modeling. *J. Clim.* **2006**, *19*, 3088–3111. [CrossRef]

21. HTCondor Manuals. Available online: https://research.cs.wisc.edu/htcondor/manual/ (accessed on 30 October 2020).
22. Thain, D.; Tannenbaum, T.; Livny, M. Distributed computing in practice: The Condor experience. *Concurr. Comput. Pract. Exp.* **2005**, *17*, 323–356. [CrossRef]