DOI: 10.1111/tgis.12901

RESEARCH ARTICLE

Transactions in GIS WILEY

# PyCLKDE: A big data-enabled high-performance computational framework for species habitat suitability modeling and mapping

**Guiming Zhang**

Department of Geography & The Environment, University of Denver, Denver, Colorado, USA

**Correspondence**
Guiming Zhang, Department of Geography & The Environment, University of Denver, 2050 E. Iliff Avenue, Denver, CO 80208, USA.
Email: guiming.zhang@du.edu

## Abstract

Species habitat suitability modeling and mapping (HSM) at large spatial scales (e.g., continental, global) and at fine spatiotemporal resolutions helps understand spatiotemporal dynamics of species distributions (e.g., migratory birds). Such HSM endeavors often involve "big" environmental and species datasets, which traditional software tools are often incapable of handling. To overcome the computational challenges facing conducting big data-involved HSM tasks, this study develops a big data-enabled high-performance computational framework to conduct HSM efficiently on large numbers of species records and massive volumes of environmental covariates. As a demonstration of its usability, PyCLKDE was implemented based on the computational framework for flexibly integrating multi-source species data for HSM. The computing performance of PyCLKDE was thoroughly evaluated through experiments modeling and mapping *Empidonax virescens* habitat suitability in the continental Americas using high-resolution environmental covariates and species observations obtained from citizen science projects. Results show that PyCLKDE can effectively exploit computing devices with varied computing capabilities (CPUs and GPUs on high-end workstations or commodity laptops) for parallel computing to accelerate HSM computations. PyCLKDE thus enables conducting big data-involved HSM using commonly available computing resources. Using PyCLKDE as an example, efforts are

  

called for to develop similar geocomputation tools based on the proposed framework to realize the potential of effectively performing geospatial big data analytics utilizing heterogenous computing resources on "personal-grade" computing resources.

## 1 | INTRODUCTION

Habitat suitability modeling and mapping (HSM) establishes mathematical models depicting species responses to environmental conditions and, based on such models, generates predictive maps representing spatial variation of species habitat suitability (Franklin & Miller, 2009). HSM, along with other similar techniques such as species distribution modeling, ecological niche modeling, gradient analysis, and resource selection functions (Hirzel & Le Lay, 2008), is widely used for both theoretical and applied purposes (Franklin & Miller, 2009). For example, the derived species–environment response models reveal species ecological niches, so resource requirements for species survival can be better understood (Soberón & Nakamura, 2009; Thorn, Nijman, Smith, & Nekaris, 2009). The produced habitat suitability maps inform decision-making in biodiversity conservation (e.g., spatial conservation prioritization) (Rodríguez, Brotons, Bustamante, & Seoane, 2007). HSM is a two-step process. In the modeling step, the relationships between species habitat suitability and environmental conditions are modeled based on localities of species occurrence and absence indicating habitat preference (or avoidance) and values of environmental covariates at these locations characterizing in-situ environmental conditions using quantitative methods (Guisan & Zimmerman, 2000). At the prediction (mapping) step, the species–environment response model is projected into geographic space to produce a gridded (raster) suitability map by applying the model to a stack of environmental covariate layers in a cell-by-cell fashion (Franklin & Miller, 2009) (i.e., the habitat suitability value at each raster cell is calculated following mathematical equations in the model with covariate values at that cell as inputs).

Understanding spatiotemporal dynamics of species distributions (e.g., migratory birds) may require HSM efforts at large spatial scales (e.g., continental, global) and/or at fine spatiotemporal resolutions (Fink et al., 2020). Traditionally, HSM studies are constrained by the scarcity of high-resolution environmental covariate data and the high costs of collecting species data (e.g., through field surveys). As a result, many studies relied on relatively small numbers of species records and were conducted in small geographic areas or at rather coarse resolutions. Recently, the increasing availability of geospatial data capturing species occurrence and environmental covariates at high spatial and temporal resolutions has made it feasible to pursue HSM involving big data (Farley, Dawson, Goring, & Williams, 2018). For instance, remote sensing technologies enable obtaining spatially and temporally high-resolution environmental data products at an unprecedented speed (Tuanmu & Jetz, 2015). Modern biodiversity-themed citizen science projects such as eBird (Wood, Sullivan, Iliff, Fink, & Kelling, 2011) and iNaturalist (Unger, Rollins, Tietz, & Dumais, 2021) engage millions of volunteer participants across the globe in contributing tens of thousands of species observations on a daily basis. Even more abundant species records are compiled and made freely available by the Global Biodiversity Information Facility (GBIF.org, 2021).

Nonetheless, methodological and computational challenges are associated with modeling and mapping species habitat suitability with such big data. Regarding methodological issues, volunteer-contributed species data are subject to various biases (e.g., spatial, temporal, contributor, and observation bias) (Zhang, 2020), which often degrade the quality of inferences drawn from the data (Zhang & Zhu, 2018). Devising methodologies to adequately mitigate the adverse impacts of biases in species data on modeling and prediction has long been an active area of research (Johnston, Moran, Musgrove, Fink, & Baillie, 2020; Pardo, Pata, Gómez, & García, 2013; Zhang &

Zhu, 2019; Zhu et al., 2015). Moreover, as species data are accumulating from many different sources, it is beneficial to develop methods for effectively integrating multi-source species data for HSM. Some methodological frameworks have been proposed and tested for integrating species data at various levels (Fletcher et al., 2019; Zhang et al., 2020).

From a computational perspective, conducting HSM over extensive geographic areas that involve large numbers of species records, voluminous environmental covariate layers, and complex modeling methods implies intensive computing demands. Existing HSM software tools, developed mainly for traditional HSM tasks that often utilize relatively small datasets, may not be capable of handling big data at all, or at least efficiently. There are a variety of software tools for conducting HSM and similar exercises (e.g., species distribution modeling) (see Ahmed et al., 2015). Many HSM tools are implemented as R packages because of the rich set of statistical methods readily available in the R environment. Some tools are implemented in the Python programming language or in Matlab (a proprietary programming language and computing environment). Other tools are also provided as a standalone graphical user interface (GUI) or cloud computing workflow (Table 1).

Two common bottlenecks exist when these tools are used to conduct big data-involved HSM tasks. The first is that existing tools are not able to handle large volumes of environmental data. The tools operate assuming that environmental covariate layers can be read into computer memory all at once and are readily available for subsequent computations (e.g., extracting covariate values at species occurrence localities for model calibrations, conducting model-based prediction cell-by-cell to produce a suitability map). This premise is reasonable for traditional HSM tasks conducted in small study areas or at coarse spatial resolutions, but becomes untenable when stacks of high-resolution environmental covariates over large areas are used in HSM tasks, as the volume of the environmental datasets can often exceed hundreds of gigabytes, if not terabytes, which is well beyond the memory space available on most computers (Zhang, Zhu, Liu, Guo, & Zhu, 2021). If the tools are run on such large datasets, they would simply halt or crash.

The second bottleneck is extremely long computing time. Most tools do not exploit data and computation parallelism and run only on a single computing thread enabled by the physical core of a central processing unit (CPU). Even if the memory bottleneck was overcome, it could take extremely long for such tools to

**TABLE 1** Existing software tools for conducting species habitat suitability modeling and mapping and similar tasks (e.g., species distribution modeling)

| Name | Form | Reference |
|---|---|---|
| sdm | R package | Naimi and Araújo (2016) |
| BIOMOD | R package | Thuiller, Lafourcade, Engler, and Araújo (2009) |
| dismo | R package | Hijmans and Elith (2013) |
| SDMTools | R package | VanDerWal et al. (2014) |
| adeHabitatHS | R package | Calenge (2015) |
| SDMToolbox | Python tools integrated within ArcGIS | Brown (2014) |
| CNN-SDMs | Python package | Deneu et al. (2021) |
| MOGP-SDM | Python package | Ingram et al. (2020) |
| Joint-SDM | Matlab extension | Tikhonov et al. (2020) |
| ENFA | Standalone GUI programmed with C/C++ | Hirzel, Hausser, Chessel, and Perrin (2002) |
| openModeller | Standalone GUI programmed with C/C++ | Muñoz et al. (2011) |
| Maxent | Standalone GUI programmed with Java | Phillips et al. (2020) |
| Cloud-SDM | Cloud computing workflow | Candela et al. (2016) |

conduct big data-involved HSM computing tasks, wherein large amounts of species data and complex modeling methods are used in model calibrations, and model-based prediction (mapping) is performed on stacks of high-resolution environmental covariates. To the best of my knowledge, only "Maxent" can utilize multiple CPU threads for computation (Phillips, Dudík, & Schapire, 2020) and the cloud-based workflow could exploit computing resources on multiple nodes (Candela, Castelli, Coro, Pagano, & Sinibaldi, 2016). Regarding accelerating computation through parallel computing, it has been proven that graphics processing units (GPUs) can bring a much greater speedup compared to CPUs (e.g., Tang, Feng, & Jia, 2015; Zhang, Zhu, & Huang, 2017). Yet there is no support for GPU parallel computation by HSM tools implemented in the R packages. Only the methods implemented in Python by Deneu et al. (2021) and Ingram, Vukcevic, and Golding (2020) utilized GPUs in model calibrations.

It is possible to complete big data-involved HSM tasks within an extended period of time by running existing tools on high-performance computing facilities, as they are equipped with powerful CPUs and abundant memory. However, only certain institutions can afford such advanced facilities. In general, few researchers, practitioners, and institutions have access to such "institutional-grade" computing resources, either physically or through cloud computing services. Instead, they have to rely on the limited computing capability (e.g., as provided by CPUs and GPUs) and memory space of "personal-grade" computing hardware available to them (e.g., laptop, desktop, workstation, and server computers), and effectively utilize such resources to achieve their goals via some form of high-performance computing solutions (Zhang, 2010; Zhang et al., 2021).

This study develops a big data-enabled high-performance computational framework for conducting HSM tasks that overcomes the above bottlenecks by exploiting parallel computing capabilities on "personal-grade" computers and by taking a "divide-and-conquer" strategy to avoid keeping environmental covariate layers all in memory. The framework was implemented in Python based on the widely used Geospatial Data Abstraction Library (GDAL) for raster spatial data reading/writing (Warmerdam, 2008) and the widely supported Open Computing Language (OpenCL) library for parallel computing (Stone, Gohara, & Shi, 2010). As an example, the kernel density estimation (KDE)-based method for modeling and mapping species habitat suitability (Zhang et al., 2020; Zhang, Zhu, Windels, & Qin, 2018) was implemented using the framework. The usability of the implemented tool, named PyCLKDE, was demonstrated through experiments of modeling and mapping habitat suitability for the Acadian flycatcher (*Empidonax virescens*) in the continental Americas at 1 km spatial resolution using a stack of 31 environmental covariates and species records from eBird and iNaturalist.

## 2 | METHODOLOGY

### 2.1 | Big data-enabled computational framework for HSM

A big data-enabled high-performance framework (Figure 1) was developed to overcome the two computational bottlenecks in modeling and mapping species habitat suitability involving large numbers of species localities and/or high-resolution covariates over large areas. This framework features two strengths compared to existing HSM software tools. First, covariate files are read into computer memory all at once only if the full-extent covariates can fit in the memory. Otherwise, only one block of the covariates at a time is read. Accordingly, covariate values at species localities are extracted without keeping the full-extent covariates in memory. The prediction step and subsequent writing results to files are also conducted in a block-by-block fashion. Moreover, the computationally intensive steps (e.g., modeling and prediction) are accelerated through parallel computing on CPUs and/or GPUs of any level of computing capabilities.

The framework was implemented in Python following an object-oriented programming paradigm. Python bindings of GDAL were used for block-based raster reading/writing (GDAL/OGR contributors, 2021). GDAL is a widely used open-source C/C++ library for efficiently reading and writing raster spatial data (Warmerdam, 2008)

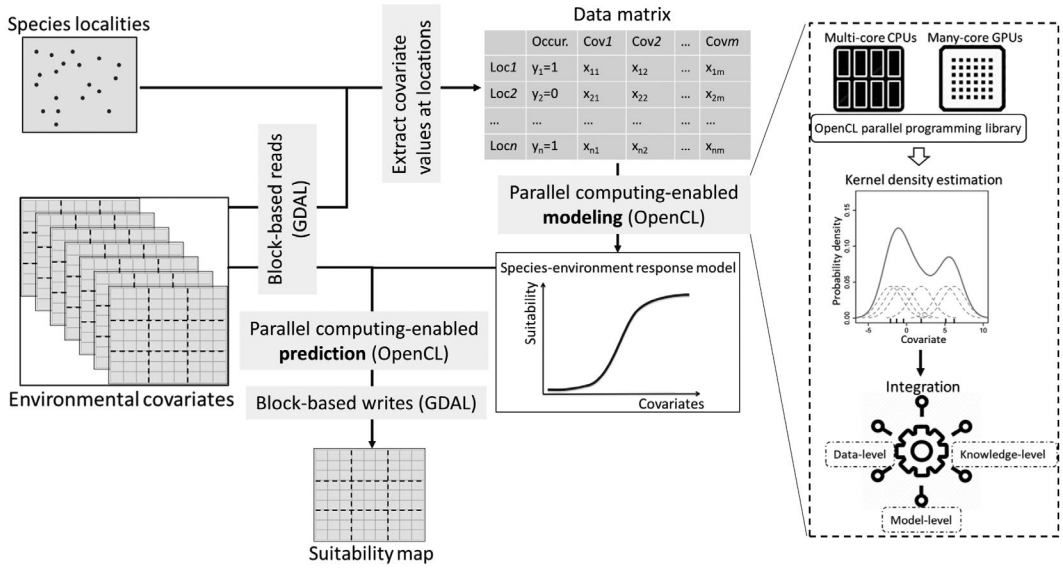


**FIGURE 1** Workflow of the big data- and parallel computing-enabled framework for species habitat suitability modeling and mapping. The dashed box on the right illustrates implementing the kernel density estimation-based method for suitability modeling (Section 2.2) using the framework

and it supports reading and processing raster by rectangular blocks with minimal overheads (Qin & Zhu, 2020). A utility class named "raster" was designed to represent the environmental covariate layer and implement block-based raster operations (e.g., reading/writing). This class can read/write data from/to GeoTIFF files. Another utility class "points" was designed to represent species locations and support relevant manipulations (e.g., extracting covariate values at species locations based on block-based raster reads). This class can read/write species locations from/to CSV files. A configuration file is offered wherein the user can configure whether to enable the "block mode" and, if yes, determine block dimension according to available computer memory, covariate data volume, and physical layout of covariate raster files (Zhang et al., 2021). The Python package NumPy (Harris et al., 2020) was used to perform various underlying array operations.

Parallelizing HSM computations was implemented based on PyOpenCL, the Python wrappers of OpenCL (Klöckner et al., 2012). OpenCL is a C/C++ parallel programming library that can exploit heterogenous parallel computing resources on any hardware (e.g., multi-core CPUs, many-core GPUs) of any level of computing capabilities (e.g., commodity computers, high-end server computers) manufactured by any vendors (e.g., NVIDIA, AMD, Intel) as long as they support the OpenCL standard (Stone et al., 2010). PyOpenCL runs parallel computation tasks on computing devices by executing a kernel function on multiple threads simultaneously (Klöckner et al., 2012; Stone et al., 2010). Only the kernel function needs to be written in C/C++. Other operations such as moving data back-and-forth between host and device are implemented in Python within the framework utilizing PyOpenCL functionalities. The host Python program compiles kernel functions on-the-fly and generates GPU/CPU runtime code for parallel execution on devices (Klöckner et al., 2012). With this framework, computationally expensive steps in HSM (e.g., modeling and prediction) can be accelerated by implementing kernel functions to carry out modeling and/or prediction in parallel (Section 2.2.2). The framework provides utility functions for detecting computing platforms and devices available on the computer, setting up the OpenCL computing environment, moving data between host and device, etc. Through the configuration file, the user can set which computing device to use for parallel computing. The user can build upon these utility functions (and the "raster" and "points" utility classes) and implement their own modeling method-specific kernel functions for HSM.

Other utility functions are also provided for pre-processing covariate data and species data (masking covariates so they share the same geographic extent and no-data area, removing species locations in no-data area, etc.), plotting maps (covariate layers, species locations, suitability map), plotting suitability–environment response curve regarding individual covariates, and conducting model performance (mapping accuracy) evaluation (e.g., plotting the receiver operating characteristic [ROC] curve and computing the area under the curve [AUC]; Hirzel, Le Lay, Helfer, Randin, & Guisan, 2006).

The infrastructure (e.g., utility classes and functions) and workflow provided by this generic framework are generally applicable for HSM and related tasks (e.g., species distribution modeling). The framework thus offers a skeleton for implementing big data- and parallel computing-enabled HSM tools. The developed framework is open-source and freely available at https://rb.gy/kijlwr. As a demonstration, the KDE-based method for HSM was implemented based on the framework (and named PyCLKDE).

## 2.2 | Implementing PyCLKDE based on the framework

### 2.2.1 | The KDE-based method for HSM

The KDE-based method for HSM is a "presence-only" method that requires only species occurrence localities along with environmental covariates for modeling species–environment responses (Zhang et al., 2018). It provides a mechanism to account for spatial sampling bias in species occurrences (Zhu et al., 2015) and offers a flexible methodological framework to integrate multi-source species data at different levels (data, knowledge, and model level) (Zhang et al., 2020). An overview of the KDE-based method is provided below. Readers interested in more details should refer to the original references.

*Basics of the KDE-based method*

The KDE-based method models species habitat suitability at a given environmental condition following a "rule-based" approach that involves modeling suitability at two levels: suitability regarding individual covariates and the overall suitability considering all covariates. At the individual covariate level, species habitat suitability–environment response regarding an environmental variable is modeled based on the probability density function (PDF) of covariate values at species occurrence locations (occurrence PDF) and the PDF of covariate values across the study area (background PDF). Given $n$ species occurrence localities and the covariate data layer representing environmental variable $x$ (e.g., elevation), covariate values at cells in which species occurrence locations fall (sample data points $x_i$) can be extracted. Then, the occurrence PDF can be estimated using KDE (Silverman, 1986):

$$\mathrm{PDF}_{\mathrm{occurrence}}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_x}K\left(\frac{x - x_i}{h_x}\right) \tag{1}$$

where $K$ is a kernel density function for which the Gaussian kernel is often adopted (Silverman, 1986). Essentially, the probability at estimation data point $x$ is the average of kernel density contributions from all sample data points $x_i$. In computing the PDF, all sample data points $x_i$, by default, are weighted with an equal weight of $1/n$. However, differing weights can be used to correct for sample selection bias in species occurrences (Zhu et al., 2015).

In Equation (1), $h_x$ is a smoothing parameter called "bandwidth." It is a crucial parameter for KDE which affects the smoothness of the estimated PDF function ("flat" vs. "spiky"). When the sample size $n$ is large, the "rule-of-thumb" method can be used to determine bandwidth based on sample size and standard deviation of the sample data points (Silverman, 1986) (this is the default bandwidth option in PyCLKDE). Otherwise, an optimal bandwidth can be determined based on the maximum likelihood criterion through cross-validation on the sample data points using the "golden section search" optimization procedure (Brunsdon, 1995). When the bandwidth is constant

across sample data points $x_i$, it is a fixed-bandwidth KDE, which tends to oversmooth the PDF over intervals with dense sample data points. In contrast, adaptive KDE—wherein the bandwidth is allowed to vary across sample data points—could reveal more subtle density variations (Zhang et al., 2017). Optimal parameters for computing adaptive bandwidths can also be determined based on the maximum likelihood criterion (Brunsdon, 1995).

Similarly, background PDF can be estimated using the KDE method. In this case, the sample data points consist of covariate values at all raster cells (or often a random subset of cells) in the study area, and sample size is the number of cells (the "rule-of-thumb" bandwidth by default is always used in PyCLKDE for estimating background PDFs, although users can change it to the other two bandwidth options through a configuration file). Based on the estimated occurrence PDF and background PDF, the ratio of the two functions, $\mathrm{PDF}_{\mathrm{ratio}}(x)$ (Equation 2), is computed to indicate habitat preference while taking resource availability into account (Zhang et al., 2018); the habitat suitability regarding covariate $x$, $S(x)$, is modeled as an inverse-logit of the ratio (Equation 3):

$$\mathrm{PDF}_{\mathrm{ratio}}(x) = \frac{\mathrm{PDF}_{\mathrm{occurence}}(x)}{\mathrm{PDF}_{\mathrm{background}}(x)} \tag{2}$$

$$S(x) = \frac{1}{1 + e^{1-\mathrm{PDF}_{\mathrm{ratio}}(x)}} \tag{3}$$

The overall suitability considering all $m$ covariates (i.e., $x^1, x^2, x^j, \ldots, x^m$) can then be determined based on the suitability values regarding individual covariates following a certain synthesizing strategy, for example, "weighted average" [Equation (4); covariates are weighted equally by default] or "limiting factor" (Zhang et al., 2018; Zhu et al., 2015):

$$S\left(x^1, x^2, x^j, \ldots, x^m\right) = \sum_{k=1}^{m} w_k S\left(x^k\right) \tag{4}$$

Conceptually, Equations (1)–(4) can be applied, in sequence, to each raster cell in the study area to compute a suitability value, and hence predict a suitability map for the study area.

*Integrating multi-source species data*

The KDE-based method also provides a framework for integrating species data from multiple sources for modeling and mapping species habitat suitability (Zhang et al., 2020). The integration can be achieved at three levels: data, knowledge, and model level. For data-level integration, species occurrence locations are simply pooled together before estimating the occurrence PDF.

For knowledge-level integration, species occurrences from different sources are used separately in modeling up to Equation (3) (computing suitability regarding individual covariates). Outputs from Equation (3) are function curves reflecting "knowledge" on species suitability–environment responses regarding individual environmental variables. The knowledge-level integration is therefore achieved by synthesizing the response curves modeled from different data sources (on each covariate) following a certain integration strategy (e.g., obtaining a synthesized curve by taking the minimum, mean, or maximum of the curves; Zhang et al., 2020).

For model-level integration, species occurrences from different sources are used separately in modeling up to Equation (4) (computing overall suitability considering all covariates). The suitability values modeled from different data sources are then synthesized following a certain integration strategy (e.g., computing the synthesized suitability as the minimum, mean, or maximum of the suitability values; Zhang et al., 2020).

The integration operators (e.g., minimum, mean, maximum) used in knowledge- and model-level integration represent different principles for data integration. While "minimum" reflects a rather conservative and stringent approach, "maximum" stands for a liberal and tolerant view. Zhang et al. (2020) reported data- and model-level integration

performed better than knowledge-level integration in terms of mapping accuracy, and using the "minimum" integration operator resulted in higher accuracy. Thus, "minimum" was set as the default integration operator. Nonetheless, the KDE-based method offers a flexible framework for integrating multi-source species data at different levels and can accommodate various integration principles. With this integration framework, one may choose the appropriate level of integration and integration strategy, depending on the specific context of the HSM task at hand.

### 2.2.2 | Implementing PyCLKDE

The KDE-based method was implemented using the proposed big data-enabled computational framework for HSM (named PyCLKDE). PyCLKDE makes use of the utility classes ("raster" and "points") and functions for reading in covariates (in "block mode" if necessary) and species locations, extracting the data matrix needed for modeling, performing model evaluation, and writing the predicted suitability map to files. The framework only needed to be extended to implement the KDE-based modeling and the prediction steps.

KDE is used for estimating PDFs in the modeling step. KDE is known to be computationally expensive when performed on large datasets (e.g., large numbers of sample data points) and/or iterative optimization procedures are used to determine the optimal bandwidths (Yuan, Chen, Gui, Li, & Wu, 2019; Zhang et al., 2017). A class named "KDE" was designed and implemented to compute probability densities (Equation 1) at many estimation data points in parallel on multi-core CPUs or many-core GPUs based on the PyOpenCL library. That is, each parallel computing thread computes the probability density at one estimation data point by calculating and averaging density contributions from all sample data points. A kernel function implementing the computation was written in C/C++ in just a few lines of code. The class has functions to manage data exchange between the host and the computing device. Moreover, it offers the flexibility of choosing different bandwidth options in estimating PDFs (i.e., "rule-of-thumb" fixed bandwidth, cross-validated fixed bandwidth, or adaptive bandwidth) through a configuration file.

Taking computation processes underlying the KDE-based method (modeling, prediction, and integration) literally, one would apply the sequence of computation steps cell-by-cell to generate a final suitability map. That is, using the covariate value at one cell as an input to compute a final suitability value before repeating the same computations at the next cell. However, this is computationally inefficient as computations may be unnecessarily repeated. For instance, the computation to determine bandwidth would be repeated when conducting KDE at each cell although the bandwidth, once determined, does not change across the cells. To avoid such repetitions, each step can be "bulk" applied to multiple inputs. As an example, when computing the occurrence PDF regarding covariate $x$ (e.g., elevation), evenly spaced estimation data points can be generated within the range of this covariate (e.g., elevation values starting from the minimum elevation to the maximum elevation at 1-m interval). KDE (Equation 1) can be applied to compute probability densities at the estimation data points in parallel with an OpenCL kernel function, needing to compute bandwidth only once. The background PDF can be computed in a similar way. A suitability–environment response function can then be derived based on the two PDFs (Equations 2 and 3). It is essentially a curve depicting how suitability changes across the gradient of this covariate (knowledge-level integration is achieved by synthesizing such curves). Next, this function can be bulk applied to the covariate data layer to transform it into a suitability data layer regarding this covariate (a cell's suitability is the value of the response function at the estimation data point closest to the cell's covariate value). Suitability layers regarding individual covariates can then be synthesized to compute an overall suitability layer (Equation 4) (model-level integration is performed by synthesizing suitability layers resulting from multiple data sources). Such bulk operations on arrays can be performed very efficiently with functionalities offered by NumPy (Harris et al., 2020), and thus were not parallelized using PyOpenCL.

A class named "HSM" was designed to implement the complete workflow of the KDE-based method for HSM, either using a single set of species data or by integrating multi-source species data. PyCLKDE is open-source and

freely available on the GitHub site. Interested parties are welcome to use PyCLKDE for HSM exercises, and to extend the big data- and parallel computing-enabled framework to implement their own modeling method-specific HSM workflow.

## 3 | EXPERIMENTS

Experiments were conducted to demonstrate the applicability of PyCLKDE to integrate multi-source data for modeling and mapping habitat suitability of *E. virescens* involving large numbers of species locations and high-resolution environmental covariates over North and South America. The computing performance of PyCLKDE, in terms of execution time, was also evaluated in different computing environments through the experiments. Execution times (e.g., total execution time, I/O time for reading covariates and writing resultant suitability raster, KDE computation time for estimating background and occurrence PDFs, and time for other computations) were recorded as the median execution times of three repeated runs.

### 3.1 | Experiment data

#### 3.1.1 | Species data

Occurrence localities of *E. virescens* were obtained from eBird and iNaturalist, two large-scale citizen science projects where birders and nature watchers around the world collaboratively contribute species sightings to biodiversity databases (Vardi, Berger-Tal, & Roll, 2021; Wood et al., 2011). *E. virescens* is a small-sized migratory bird that is present in both North and South America. A set of $n = 54{,}684$ occurrence locations (Figure 2a) were extracted from the eBird basic dataset (eBird, 2019) over a 10-year period (2010–2019). Another $n = 211$ occurrences (Figure 2b) over the same time period were extracted from the iNaturalist "research-grade" observations dataset (Ueda, 2021). There are many more *E. virescens* observations from eBird as birders in general prefer eBird over iNaturalist for logging birding records. Nonetheless, *E. virescens* occurrences from the two sources were used in the experiments to illustrate multi-source species data integration for modeling and mapping species habitat suitability.

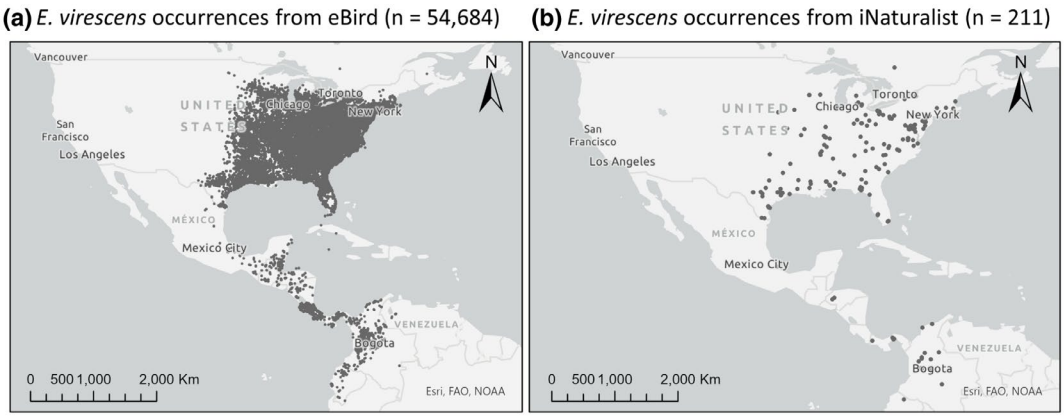


FIGURE 2 Occurrence locations of *E. virescens* obtained from: (a) eBird; and (b) iNaturalist (2010–2019)

**TABLE 2** Specifications of the two computing environments used to test PyCLKDE

| Computing environment | Operating system | CPU | GPU |
|---|---|---|---|
| Desktop: Dell Precision 5820 | Windows 10 | Intel Xeon CPU (8 cores, 16 logical processors)<br>Max clock speed: 3.7 GHz<br>Memory: 64 GB | NVIDIA Quadro P4000 (discrete)<br>Max clock speed: 1.5 GHz<br>Memory: 8 GB |
| Laptop: Dell Inspiron 14 | Windows 10 | Intel i–7 (2 cores, 4 logical processors)<br>Max clock speed: 2.6 GHz.<br>Memory: 16 GB | 1. AMD Radeon R7 M260 (discrete)<br>Max clock speed: 0.9 GHz<br>Memory: 2 GB<br>2. Intel HD Graphics 4400 (integrated)<br>Max clock speed: 1.1 GHz<br>Memory: 1.5 GB |

### 3.1.2 | Environmental covariates

A set of 31 covariates at 1 km spatial resolution depicting bioclimatic variables, topographic variations, land cover, and terrestrial habitat heterogeneity were used for modeling and mapping *E. virescens* habitat suitability. Among them, 19 biologically meaningful bioclimatic variables were downloaded from WorldClim at 1 km spatial resolution (Fick & Hijmans, 2017). These variables represent annual trends (e.g., mean annual temperature, annual precipitation), seasonality (e.g., annual range in temperature and precipitation), and extreme or limiting environmental factors (e.g., temperature of the coldest and warmest month, and precipitation of the wet and dry quarters), and are often used in species distribution modeling and related ecological modeling techniques. Topographic data were extracted from the 1 km resolution global topography dataset (Amatulli et al., 2018). Elevation, slope gradient, sine of aspect, cosine of aspect, and roughness were used to reflect terrain variations. A land cover class covariate was derived from the 1 km global consensus land-cover product (evergreen/deciduous needleleaf trees, evergreen broadleaf trees, deciduous broadleaf trees, mixed/other trees, shrubs, herbaceous vegetation, cultivated and managed vegetation, regularly flooded vegetation, urban/built-up, snow/ice, barren, and open water; Tuanmu & Jetz, 2014). A set of six variables was adopted to characterize terrestrial habitat heterogeneity at 1 km resolution (coefficient of variation, evenness, range, Shannon diversity index, Simpson diversity index, and standard deviation; Tuanmu & Jetz, 2015). They are first-order texture measures derived from Enhanced Vegetation Index imagery acquired by the Moderate Resolution Imaging Spectroradiometer (MODIS).

All covariates (except for land cover) were normalized by first subtracting their respective mean from the original cell value and subsequently dividing the difference by their respective standard deviation. Covariate layers were stored as individual GeoTIFF files and were all masked to the spatial extent of the study area. The storage size of the files was roughly 40.4 GB. Each covariate layer is of the same dimension, 16,738 rows × 20,011 columns, and is stored in the GeoTIFF file following a 1,024 rows × 1,216 columns block layout. If a covariate layer is read into memory in its full extent, it would require ~2.5 GB of memory space (Python uses 8 bytes to represent real numbers). Reading all 31 covariates into memory would need ~77.5 GB of memory, which outsizes the memory space on most computers. Thus, covariates had to be read into computer memory block by block at a certain block size. Block dimensions were set to multiples of the physical file block size (e.g., 2,048 rows × 1,216 columns) in PyCLKDE to improve reading speed (Qin & Zhu, 2020; Zhang et al., 2021). Based on sensitivity analyses on block dimension settings (Section 4.4), the block dimension for the experiments was set to 4,096 rows × 4,864 columns and 1,024 rows × 1,216 columns on the desktop and laptop, respectively.

## 3.2 | Computing environments

PyCLKDE was tested in two computing environments with distinct hardware and computing capabilities (Table 2). The high-end desktop workstation is equipped with an 8-core Intel Xeon CPU and an NVIDIA GPU. The commodity laptop computer has a 2-core Intel i-7 CPU and a discrete AMD GPU as well as an integrated Intel GPU. The computing environments were used to demonstrate the compatibility of PyCLKDE across computing hardware manufactured by different vendors with varying computing capabilities, and to assess how these varying conditions would impact the computing performance of PyCLKDE. When running PyCLKDE in the two computing environments, the block dimension for reading covariates was set to 4,096 rows × 4,864 columns for the desktop and 1,024 rows × 1,216 columns for the laptop.

## 3.3 | Experiment results

### 3.3.1 | Habitat suitability maps

The KDE-based method was applied to model and map *E. virescens* habitat suitability based on species occurrences from only eBird and by integrating species occurrences from eBird and iNaturalist at the data, knowledge, and model level (Figure 3). The training AUC values (computed with occurrence locations used in modeling plus randomly selected background locations) were all above 0.75 and close, indicating satisfactory model fitting (Hirzel et al., 2006). Compared to modeling using species occurrences from only eBird, integrating species occurrences from iNaturalist with eBird occurrences did not significantly change the spatial pattern on the predicted suitability map. This was expected because a much smaller number of species occurrences was taken from iNaturalist. Overall, areas predicted to be of relatively high habitat suitability are the eastern U.S. (known distribution range of *E. virescens*) and the southeastern part of South America (potential distribution area).

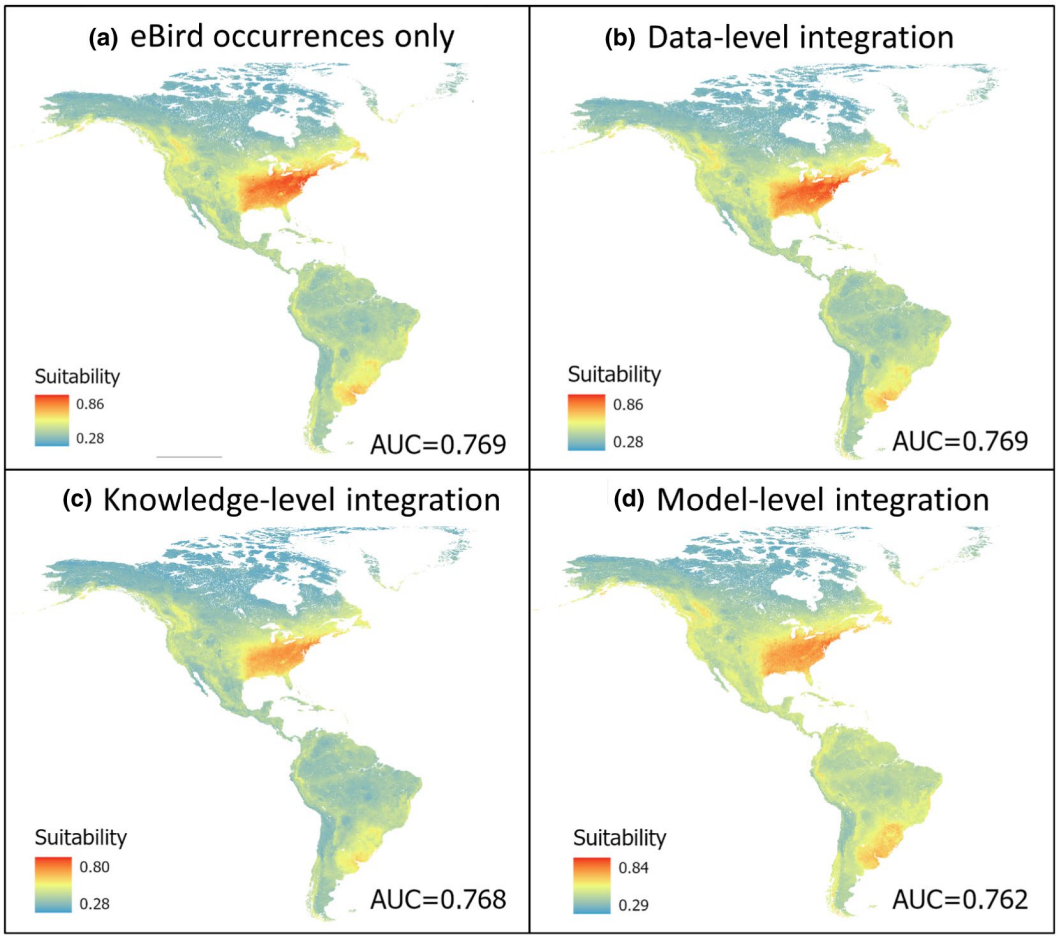


**FIGURE 3** Habitat suitability maps of *E. virescens* predicted based on occurrences from eBird only (a) and through integrating occurrences from eBird and iNaturalist at data level (b), knowledge level (c), and model level (d)

### 3.3.2 | Computing performance

The above results were obtained by running PyCLKDE with "rule-of-thumb" fixed bandwidth on the desktop workstation using NVIDIA GPU as computing device. Individual habitat suitability modeling and mapping tasks were completed within 8 min (Table 3). Among the tasks, integrating occurrence data from eBird and iNaturalist for modeling and mapping at the model level is most time-consuming as it involves more computations (e.g., reading in covariates more than once, estimating occurrence PDFs separately for each data source). Overall, computations involving using KDE to estimate background and occurrence PDFs takes up less than 2% of the total execution time as these computationally expensive steps were drastically accelerated through parallel computing (see Section 4.1 for more discussion). I/O (reading/writing) takes up about 20–30% of the total execution time, and other computations take up around 68–77%.

## 4 | DISCUSSION

### 4.1 | Effectiveness of parallel computing

The effectiveness of parallel computing in PyCLKDE was evaluated by comparing its computing performance against a benchmark where KDE computations were implemented in native Python codes that run on a single CPU thread (i.e., without OpenCL-based parallel computing). It took up to 45 min for the non-parallel version of PyCLKDE to complete individual HSM tasks (Table 4). The expensive KDE computations take up 80–86% of the total execution time. Time spent on I/O takes up less than 5%, and other computations take up 10–17%.

With respect to total execution time, PyCLKDE with OpenCL-based parallel computing running on GPU (Table 3) is overall 6 to 8 times faster than the non-parallel benchmark implementation. Nonetheless, when only comparing time spent on KDE computations, OpenCL-based parallel computing is 275 to 405 times faster. The computing performance of PyCLKDE with OpenCL-based parallel computing running on a multi-core CPU (Table 5) is similar to that running on a GPU. It is 6 to 8 times faster and 275 to 345 times faster than the benchmark in terms of total execution time and KDE computation time, respectively. OpenCL-based parallel computing significantly accelerated KDE computations in PyCLKDE and therefore could effectively speed up species habitat suitability modeling and mapping tasks involving large numbers of species occurrences.

### 4.2 | Effects of bandwidth option

The effects of KDE bandwidth option (Section 2.2.1.1) were examined by running PyCLKDE with three different bandwidth options in estimating occurrence PDFs (i.e., "rule-of-thumb" fixed bandwidth, cross-validated fixed

**TABLE 3** Execution time (s) of PyCLKDE on the desktop workstation using NVIDIA GPU as computing device

| | | I/O | | KDE | | |
|---|---|---|---|---|---|---|
| | Total | Read | Write | Background | Occurrence | Other |
| eBird occurrences only | 317.03 | 39.65 | 57.00 | 2.66 | 2.57 | 215.15 |
| Data-level integration | 315.48 | 38.96 | 57.01 | 2.60 | 2.60 | 214.31 |
| Knowledge-level integration | 389.92 | 38.89 | 57.72 | 2.53 | 4.81 | 285.97 |
| Model-level integration | 445.52 | 39.11 | 56.33 | 2.60 | 5.00 | 342.47 |

**TABLE 4** Execution time (s) of PyCLKDE on the desktop workstation with KDE computations implemented in native Python

| | | I/O | | KDE | | |
|---|---|---|---|---|---|---|
| | Total | Read | Write | Background | Occurrence | Other |
| eBird occurrences only | 2,465.36 | 40.74 | 57.82 | 945.72 | 1,169.49 | 251.58 |
| Data-level integration | 2,542.58 | 40.59 | 57.27 | 931.19 | 1,206.90 | 306.64 |
| Knowledge-level integration | 2,630.23 | 40.33 | 56.88 | 928.81 | 1,207.07 | 397.14 |
| Model-level integration | 2,625.18 | 40.92 | 56.67 | 922.67 | 1,172.12 | 432.81 |

**TABLE 5** Execution time (s) of PyCLKDE on the desktop workstation with OpenCL-based parallel computing on CPU

| | | I/O | | KDE | | |
|---|---|---|---|---|---|---|
| | Total | Read | Write | Background | Occurrence | Other |
| eBird occurrences only | 313.72 | 38.79 | 56.90 | 2.95 | 3.16 | 211.91 |
| Data-level integration | 313.43 | 38.93 | 56.17 | 2.97 | 3.25 | 212.12 |
| Knowledge-level integration | 386.55 | 38.92 | 56.35 | 2.93 | 4.67 | 283.68 |
| Model-level integration | 442.16 | 39.21 | 57.04 | 2.92 | 4.72 | 338.27 |

**TABLE 6** Execution time (s) of PyCLKDE with different bandwidth options running on the desktop workstation using NVIDIA GPU as computing device

| | | I/O | | KDE | | |
|---|---|---|---|---|---|---|
| | Total | Read | Write | Background | Occurrence | Other |
| Fixed bandwidth ("rule-of-thumb") | 445.52 | 39.11 | 56.33 | 2.60 | 5.00 | 342.47 |
| Fixed bandwidth (cross-validation) | 521.29 | 39.26 | 57.48 | 2.76 | 81.83 | 339.95 |
| Adaptive bandwidth (cross-validation) | 522.71 | 39.19 | 57.37 | 2.79 | 83.32 | 340.04 |

bandwidth, and cross-validated adaptive bandwidth) for modeling and mapping habitat suitability of *E. virescens* by integrating occurrences from eBird and iNaturalist at the model level. The major effect of bandwidth option on the computing performance of PyCLKDE lies in the time spent on estimating occurrence PDFs using KDE (Table 6). On the experiment dataset, estimating occurrence PDFs with cross-validated fixed bandwidth and adaptive bandwidth is roughly 16 times more time-consuming compared to estimating PDFs with "rule-of-thumb" bandwidth, as the optimization procedure used to determine optimal bandwidth based on cross-validation involves multiple iterations of KDE computations.

The KDE bandwidth option also affects estimated occurrence PDFs (and hence modeling and mapping results). The smoothness of estimated occurrence PDFs differs across three bandwidth options (Figure 4). Compared to "rule-of-thumb" bandwidth, using cross-validated bandwidth (fixed or adaptive) reveals more subtle density variations along environmental gradients (e.g., mean annual temperature). PDFs estimated with cross-validated bandwidth thus could potentially offer more detailed ecological insights on how species

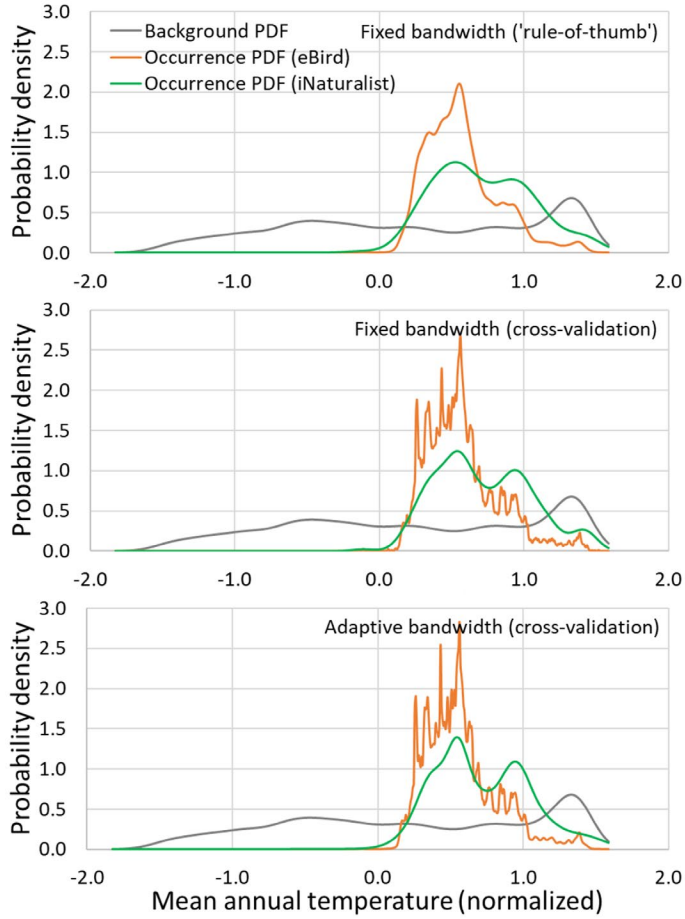


**FIGURE 4** Occurrence PDFs regarding mean annual temperature estimated based on occurrence locations from eBird and iNaturalist using three different bandwidth options

respond to environmental conditions at higher resolution (e.g., PDFs estimated based on occurrences from iNaturalist). However, they also risk amplifying noise in the sample data (e.g., PDFs estimated based on occurrences from eBird are overly "spiky"). In contrast, PDFs estimated with "rule-of-thumb" bandwidth are more robust against noise and thus could uncover generalized species–environment response curves. PyCLKDE uses "rule-of-thumb" bandwidth by default for estimating occurrence PDFs, while users can switch to the other two bandwidth options. One should choose the proper bandwidth option based on the specific context of the HSM task at hand.

## 4.3 | Compatibility with computing devices

PyCLKDE was run on different computing devices in the two computing environments (Table 2) to assess its compatibility with various computing devices and their impacts on computing performance of PyCLKDE. On the desktop, running PyCLKDE on GPU (Table 3) to integrate species occurrences from eBird and iNaturalist at the model level for modeling and mapping habitat suitability took about the same amount of time as running it on

a multi-core CPU (i.e., ~6 min) (Table 5). They are both ~7 times faster and over 275 times faster than the non-parallel benchmark in terms of total execution time and KDE computation time, respectively.

Running PyCLKDE on the laptop to complete the same HSM task (Table 7) is slower compared to the desktop, which is as expected given the desktop is equipped with more advanced CPU and GPU (e.g., higher max clock speeds). On the laptop, it took 75 min with the non-parallel benchmark, and 80% of the total execution time (i.e., 60 min) was spent on KDE computations. Using PyCLKDE with OpenCL-based parallel computing on a multi-core CPU, discrete GPU, or integrated GPU, it took about 13 min (i.e., 6 times faster than the non-parallel benchmark) and less than 2% of the total execution time was spent on KDE computations. Only comparing KDE computation time against the non-parallel benchmark, KDE computations with parallel computing on a CPU, discrete GPU, and integrated GPU are 237 times, 619 times, and 543 times faster, respectively.

Clearly, PyCLKDE is able to exploit computing environments and computing devices of varying computing capabilities to significantly speed up KDE computations and thus enable HSM tasks involving a large number of species occurrences.

## 4.4 | Impacts of block dimension

PyCLKDE was run with different settings of block dimension by which covariates are read into computer memory to examine how block dimension setting affects I/O time. The tested block dimensions were obtained by multiplying an integer number (e.g., 1, 2, 4, 6, 8, and 10) by the block dimension of the physical GeoTIFF files (1,024 rows × 1,216 columns). Overall, time spent on reading covariates increases with larger blocks whilst time spent on writing the resultant suitability map decreases (Table 8). With respect to the combined I/O time on the desktop,

**TABLE 7** Execution time (s) of PyCLKDE on the laptop for modeling and mapping habitat suitability by integrating species occurrences from eBird and iNaturalist at the model level

| | | I/O | | KDE | | |
|---|---|---|---|---|---|---|
| | Total | Read | Write | Background | Occurrence | Other |
| Benchmark (non-parallel) | 4,510.38 | 60.26 | 139.02 | 1,614.71 | 2,027.82 | 668.57 |
| CPU parallel | 803.29 | 55.28 | 137.78 | 5.78 | 9.58 | 594.88 |
| GPU (discrete) parallel | 759.55 | 48.02 | 142.04 | 2.27 | 3.61 | 563.62 |
| GPU (integrated) parallel | 763.23 | 59.41 | 137.49 | 3.20 | 3.50 | 559.63 |

**TABLE 8** Execution time of PyCLKDE with different block dimension settings running on the desktop using NVIDIA GPU as computing device. PyCLKDE was applied for modeling and mapping habitat suitability by integrating species occurrences from eBird and iNaturalist at the model level

| Block dimension (rows × columns) | | I/O | | KDE | | |
|---|---|---|---|---|---|---|
| | Total | Read | Write | Background | Occurrence | Other |
| 1024 × 1216 | 678.54 | 67.32 | 173.51 | 2.83 | 4.94 | 429.94 |
| 2048 × 2432 | 479.82 | 38.86 | 95.71 | 2.85 | 5.48 | 336.93 |
| 4096 × 4864 | 445.52 | 39.11 | 56.33 | 2.60 | 5.00 | 342.47 |
| 6144 × 7296 | 619.66 | 215.61 | 33.95 | 2.63 | 4.68 | 362.79 |
| 8192 × 9728 | 703.94 | 283.02 | 37.21 | 2.55 | 4.79 | 376.36 |
| 10,240 × 12,160 | 838.32 | 350.97 | 24.14 | 2.83 | 4.64 | 455.74 |

the optimal block dimension was 4,096 rows × 4,864 columns (corresponding to a multiplier of 4). With this block dimension setting, the total execution time (445.5 s) and I/O time (95.4 s) and its percentage relative to total execution time (21.4%) were both minimal. On the laptop, the optimal block dimension was 1,024 rows × 1,216 columns, with which the total execution time (759.6 s) and I/O time (190 s) and its percentage (25%) were minimal (using the discrete GPU as computing device). Throughout the experiments, PyCLKDE was run with the two optimal block dimensions on the desktop and laptop, respectively.

## 4.5 | Scalability to larger HSM tasks

Additional datasets were generated to evaluate the scalability of PyCLKDE to HSM tasks involving larger datasets. Besides the eBird occurrence dataset containing $n = 54{,}684$ occurrence locations, two larger artificial occurrence datasets with $n = 11{,}000$ occurrences and $n = 22{,}000$ occurrences, respectively, were obtained by generating random point locations in the study area. These two artificial occurrence datasets were used as species occurrences in PyCLKDE purely for computing performance evaluation purposes. Moreover, the original covariates at 1 km spatial resolution were resampled to 500 m resolution (with the 1,024 rows × 1,216 columns file block layout maintained). The storage size of the 500 m resolution GeoTIFF files was roughly 152 GB. PyCLKDE was run on the desktop using GPU as computing device to model and map habitat suitability using each of the three occurrence datasets on each of the two covariate datasets (Table 9).

On the same covariate dataset, doubling the number of species occurrences increases the total execution time by 1.5- to 2-fold, mainly through increased time on "other" computations (I/O time and KDE computation time did not change much). Covariate data size has a much more significant impact on execution times. Across the occurrence datasets, when spatial resolution improves from 1 km to 500 m (the number of cells increases 4-fold), total execution time increases by a factor of 8–12, mainly through increasing combined I/O time by a factor of 9–15 (time on reading and writing increased by a factor of about 40 and 6, respectively) and time on "other" computations by a factor of 6–10. Overall, the total execution time of PyCLKDE increases in a linear fashion on HSM tasks involving larger numbers of species occurrences or number of covariate cells.

**TABLE 9** Execution time of PyCLKDE on HSM tasks involving larger numbers of species occurrences and higher-resolution environmental covariates. Experiments were run on the desktop workstation using NVIDIA GPU as computing device to model and map habitat suitability based on a single occurrence dataset

| | | I/O | | KDE | | |
|---|---|---|---|---|---|---|
| Number of occurrences | Total | Read | Write | Background | Occurrence | Other |
| 1 km resolution covariates | | | | | | |
| 54,684 | 317.03 | 39.65 | 57.00 | 2.66 | 2.57 | 215.15 |
| 110,000 | 469.52 | 40.52 | 76.81 | 3.09 | 3.61 | 345.50 |
| 220,000 | 915.95 | 39.84 | 69.50 | 3.09 | 4.28 | 799.25 |
| 500 m resolution covariates | | | | | | |
| 54,684 | 3,831.73 | 1,674.67 | 350.49 | 3.02 | 3.03 | 1,800.53 |
| 110,000 | 5,617.38 | 1,613.72 | 438.44 | 3.01 | 3.46 | 3,558.75 |
| 220,000 | 6,976.04 | 1,624.10 | 421.02 | 3.13 | 4.38 | 4,923.42 |

### 4.6 | Bottlenecks and potential improvements

As KDE computations are effectively accelerated through OpenCL-based parallel computing, the bottlenecks of PyCLKDE computing performance are I/O (~30% of total execution time) and "other" computations (~70% of total execution time) (Section 4.1). To speed up I/O, covariates could be read into memory in parallel with multiple threads, for example, using multiprocessing implemented in the *Pathos* Python package (https://pypi.org/project/pathos/). However, multi-thread reading incurs additional computational cost for maintaining the pool of parallel threads and, as a result, the time-saving by multi-thread reading may not be as significant, especially when a proper block dimension is already exploited to speed up reading (Section 4.4) (Zhang et al., 2021). Another potential improvement is to accelerate computations in the "other" category through parallel computation. Such computations in PyCLKDE are mostly array-involved computations and currently are implemented heavily relying on functionalities offered by the NumPy package, which is highly optimized for efficient manipulations on vectors, arrays, and matrices (Harris et al., 2020). Conducting these computations using customized OpenCL-based parallel computing that can be implemented based on the proposed computational framework (Section 2.1) may further speed up the computations, especially on very large datasets.

### 4.7 | Geospatial applications of GPU computing

As revealed by the experiments, parallel computing utilizing GPUs can greatly speed up PyCLKDE for habitat suitability modeling and mapping. Generally, GPU-based parallel computing could achieve much more significant acceleration compared to parallel computing utilizing multi-core CPUs, especially on large-size computational problems (Zhang et al., 2017, 2021). The reason is that, in contrast to CPUs that emphasize improving computing power on a small number of cores (and threads), many-core GPUs offer large numbers of simpler processors (e.g., thousands or more) for massive parallelism. GPU-based parallel computing can effectively exploit data and/or task parallelism and speed up geospatial analysis and modeling tasks (Tang et al., 2015; Tang & Wang, 2020; Zhang et al., 2017). The developed PyCLKDE framework contributes to advocating the application of GPU computing in geospatial domains to enable tackling data- and computation-intensive problems that were previously intractable.

## 5 | CONCLUSIONS

To overcome the computational challenges facing big data-involved HSM tasks, this study develops a big data-enabled high-performance computational framework for efficiently conducting HSM on large numbers of species records and massive volumes of environmental covariates. Utilizing the block-based I/O mechanism, OpenCL-backed parallel computing, and other infrastructure offered by the computational framework, PyCLKDE was implemented for flexibly integrating multi-source species data to model and map species habitat suitability. The computing performance of PyCLKDE was thoroughly evaluated through a case study of modeling and mapping *E. virescens* habitat suitability in the continental Americas using high-resolution environmental covariates and species observations obtained from two citizen science projects. Experiment results show that PyCLKDE can effectively exploit computing devices with varied computing capabilities, such as multi-core CPUs and discrete/integrated GPUs on high-end workstations or "personal-grade" laptops, for parallel computing to accelerate HSM computations. With the support of PyCLKDE, rapidly conducting HSM at large spatial scales (e.g., continental, global) and at fine spatiotemporal resolutions is feasible, utilizing only commonly available computing resources.

Like PyCLKDE, other HSM modeling algorithms and spatial modeling and prediction methods can be implemented based on the utilities offered by the proposed computational framework. For example, digital soil mapping (DSM) is similar to HSM from a computational perspective as they both require environmental covariates and field samples

(species observations or soil samples) as inputs, based on which the relationship between environmental gradients and a target variable (habitat suitability or soil property) is modeled and applied for prediction (mapping). DSM algorithms can thus be implemented based on the computational framework to support DSM efforts involving big data (Zhang et al., 2021). With these motivating examples, efforts are called for to develop similar geocomputation tools based on the proposed framework to realize the potential of effectively performing geospatial big data analytics utilizing heterogenous computing resources on "personal-grade" computing resources (Zhang, 2010).

## ACKNOWLEDGMENTS

This research was partially supported by the Faculty Research Fund, Internationalization Grant, and Faculty Start-up Funds at the University of Denver. The author thanks participants of the eBird and iNaturalist citizen science projects for contributing species observations and making the data freely available for research.

## CONFLICT OF INTEREST

The author has no conflict of interest to declare.

## DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available through public sources as cited in the main text.

## ORCID

*Guiming Zhang* https://orcid.org/0000-0001-7064-2138

## REFERENCES

Ahmed, S. E., McInerny, G., O'Hara, K., Harper, R., Salido, L., Emmott, S., & Joppa, L. N. (2015). Scientists and software: Surveying the species distribution modelling community. *Diversity and Distributions*, *21*, 258–267. https://doi.org/10.1111/ddi.12305

Amatulli, G., Domisch, S., Tuanmu, M.-N., Parmentier, B., Ranipeta, A., Malczyk, J., & Jetz, W. (2018). A suite of global, cross-scale topographic variables for environmental and biodiversity modeling. *Scientific Data*, *5*, 180040. https://doi.org/10.1038/sdata.2018.40

Brown, J. L. (2014). SDMtoolbox: A python-based GIS toolkit for landscape genetic, biogeographic and species distribution model analyses. *Methods in Ecology and Evolution*, *5*, 694–700. https://doi.org/10.1111/2041-210X.12200

Brunsdon, C. (1995). Estimating probability surfaces for geographical point data: An adaptive kernel algorithm. *Computers and Geosciences*, *21*, 877–894. https://doi.org/10.1016/0098-3004(95)00020-9

Calenge, C. (2015). *adehabitatHS: Analysis of habitat selection by animals* (R Package). Retrieved from https://cran.r-project.org/web/packages/adehabitatHS/adehabitatHS.pdf

Candela L., Castelli D., Coro G., Pagano P., & Sinibaldi F. (2016). Species distribution modeling in the cloud. *Concurrency and Computation: Practice and Experience*, *28*, (4), 1056–1079. http://dx.doi.org/10.1002/cpe.3030

de Souza Muñoz, M. E., De Giovanni, R., de Siqueira, M. F., Sutton, T., Brewer, P., Pereira, R. S., … Canhos, V. P. (2011). openModeller: A generic approach to species' potential distribution modelling. *Geoinformatica*, *15*, 111–135. https://doi.org/10.1007/s10707-009-0090-7

Deneu, B., Servajean, M., Bonnet, P., Botella, C., Munoz, F., & Joly, A. (2021). Convolutional neural networks improve species distribution modelling by capturing the spatial structure of the environment. *PLoS Computational Biology*, *17*(4), e1008856. https://doi.org/10.1371/journal.pcbi.1008856

eBird. (2019). *eBird basic dataset metadata* (v1.12). Retrieved from https://ebird.org/science/use-ebird-data/download-ebird-data-products

Farley, S. S., Dawson, A., Goring, S. J., & Williams, J. W. (2018). Situating ecology as a big-data science: Current advances, challenges, and solutions. *BioScience*, *68*, 563–576. https://doi.org/10.1093/biosci/biy068

Fick, S., & Hijmans, R. (2017). WorldClim 2: New 1-km spatial resolution climate surfaces for global land areas. *International Journal of Climatology*, *37*, 4302–4315. https://doi.org/10.1002/joc.5086

Fink, D., Auer, T., Johnston, A., Ruiz-Gutierrez, V., Hochachka, W. M., & Kelling, S. (2020). Modeling avian full annual cycle distribution and population trends with citizen science data. *Ecological Applications*, *30*(3), e02056. https://doi.org/10.1002/eap.2056

Fletcher, R. J., Hefley, T. J., Robertson, E. P., Zuckerberg, B., McCleery, R. A., & Dorazio, R. M. (2019). A practical guide for combining data to model species distributions. *Ecology*, *100*(6), e02710. https://doi.org/10.1002/ecy.2710

Franklin, J., & Miller, J. A. (2009). *Mapping species distributions: Spatial inference and prediction*. Cambridge, UK: Cambridge University Press.

GBIF.org. (2021). *GBIF Home Page*. Retrieved from https://www.gbif.org

GDAL/OGR contributors. (2021). *GDAL/OGR Geospatial Data Abstraction Software Library*. Retrieved from https://gdal.org/

Guisan, A., & Zimmerman, N. E. (2000). Predictive habitat distribution models in ecology. *Ecological Modelling*, *135*, 147–186. https://doi.org/10.1016/S0304-3800(00)00354-9

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hijmans, R. J., & Elith, J. (2013). *Species distribution modeling with R*. Retrieved from http://www2.uaem.mx/r-mirror/web/packages/dismo/vignettes/sdm.pdf

Hirzel, A. H., Hausser, J., Chessel, D., & Perrin, N. (2002). Ecological-niche factor analysis: How to compute habitat-suitability maps without absence data? *Ecology*, *83*, 2027–2036. https://doi.org/10.1890/0012-9658(2002)083[2027:ENFAHT]2.0.CO;2

Hirzel, A. H., & Le Lay, G. (2008). Habitat suitability modelling and niche theory. *Journal of Applied Ecology*, *45*, 1372–1381. https://doi.org/10.1111/j.1365-2664.2008.01524.x

Hirzel, A. H., Le Lay, G., Helfer, V., Randin, C., & Guisan, A. (2006). Evaluating the ability of habitat suitability models to predict species presences. *Ecological Modelling*, *199*, 142–152. https://doi.org/10.1016/j.ecolmodel.2006.05.017

Ingram, M., Vukcevic, D., & Golding, N. (2020). Multi-output Gaussian processes for species distribution modelling. *Methods in Ecology and Evolution*, *11*, 1587–1598. https://doi.org/10.1111/2041-210X.13496

Johnston, A., Moran, N., Musgrove, A., Fink, D., & Baillie, S. R. (2020). Estimating species distributions from spatially biased citizen science data. *Ecological Modelling*, *422*, 108927. https://doi.org/10.1016/j.ecolmodel.2019.108927

Klöckner, A., Pinto, N., Lee, Y., Catanzaro, B., Ivanov, P., & Fasih, A. (2012). PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation. *Parallel Computing*, *38*, 157–174. https://doi.org/10.1016/j.parco.2011.09.001

Naimi, B., & Araújo, M. B. (2016). Sdm: A reproducible and extensible R platform for species distribution modelling. *Ecography*, *39*, 368–375. https://doi.org/10.1111/ecog.01881

Pardo, I., Pata, M. P., Gómez, D., & García, M. B. (2013). A novel method to handle the effect of uneven sampling effort in biodiversity databases. *PLoS One*, *8*, e52786. https://doi.org/10.1371/journal.pone.0052786

Phillips, S. J., Dudík, M., & Schapire, R. E. (2020). *Maxent software for modeling species niches and distributions* (Version 3.4.0). Retrieved from http://biodiversityinformatics.amnh.org/open_source/maxent/

Qin, C.-Z., & Zhu, L.-J. (2020). GDAL/OGR and geospatial data IO libraries. In J. P. Wilson (Ed.), *The geographic information science & technology body of knowledge*. Washington, DC: University Consortium for Geographic Information Science. Retrieved from https://gistbok.ucgis.org/bok-topics/gdalogr-and-geospatial-data-io-libraries

Rodríguez, J. P., Brotons, L., Bustamante, J., & Seoane, J. (2007). The application of predictive modelling of species distribution to biodiversity conservation. *Diversity and Distributions*, *13*, 243–251. https://doi.org/10.1111/j.1472-4642.2007.00356.x

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. London, UK: Chapman & Hall.

Soberón, J., & Nakamura, M. (2009). Niches and distributional areas: Concepts, methods, and assumptions. *Proceedings of the National Academy of Science of the United States of America*, *106*, 19644–19650. https://doi.org/10.1073/pnas.0901637106

Stone, J. E., Gohara, D., & Shi, G. (2010). OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in Science and Engineering*, *12*, 199–200. https://doi.org/10.1109/MCSE.2010.69

Tang, W., Feng, W., & Jia, M. (2015). Massively parallel spatial point pattern analysis: Ripley's K function accelerated using graphics processing units. *International Journal of Geographical Information Science*, *29*, 412–439. https://doi.org/10.1080/13658816.2014.976569

Tang, W., & Wang, S. (Eds.). (2020). *High performance computing for geospatial applications*. Berlin, Germany: Springer.

Thorn, J. S., Nijman, V., Smith, D., & Nekaris, K. A. I. (2009). Ecological niche modelling as a technique for assessing threats and setting conservation priorities for Asian slow lorises (Primates: *Nycticebus*). *Diversity and Distributions*, *15*, 289–298.

Thuiller, W., Lafourcade, B., Engler, R., & Araújo, M. B. (2009). BIOMOD: A platform for ensemble forecasting of species distributions. *Ecography*, *32*, 369–373. https://doi.org/10.1111/j.1600-0587.2008.05742.x

Tikhonov, G., Duan, L., Abrego, N., Newell, G., White, M., Dunson, D., & Ovaskainen, O. (2020). Computationally efficient joint species distribution modeling of big spatial data. *Ecology*, *101*(2), e02929. https://doi.org/10.1002/ecy.2929

Tuanmu, M. N., & Jetz, W. (2014). A global 1-km consensus land-cover product for biodiversity and ecosystem modelling. *Global Ecology and Biogeography*, *23*, 1031–1045. https://doi.org/10.1111/geb.12182

Tuanmu, M. N., & Jetz, W. (2015). A global, remote sensing-based characterization of terrestrial habitat heterogeneity for biodiversity and ecosystem modelling. *Global Ecology and Biogeography*, *24*, 1329–1339. https://doi.org/10.1111/geb.12365

Ueda, K. (2021). *iNaturalist research-grade observations: iNaturalist.org; Occurrence dataset*. Retrieved from https://doi.org/10.15468/ab3s5x accessed via GBIF.org

Unger, S., Rollins, M., Tietz, A., & Dumais, H. (2021). iNaturalist as an engaging tool for identifying organisms in outdoor activities. *Journal of Biological Education*, *55*(5), 537–547. https://doi.org/10.1080/00219266.2020.1739114

VanDerWal, J., Falconi, L., Januchowski, S., Shoo, L., Storlie, C., & VanDerWal, M. J. (2014). *Package "SDMTools"*. Retrieved from https://www.rdocumentation.org/packages/SDMTools/versions/1.1-221

Vardi, R., Berger-Tal, O., & Roll, U. (2021). iNaturalist insights illuminate COVID-19 effects on large mammals in urban centers. *Biological Conservation*, *254*, 108953. https://doi.org/10.1016/j.biocon.2021.108953

Warmerdam, F. (2008). The geospatial data abstraction library. In G. B. Hall & M. G. Leahy (Eds.), *Open source approaches in spatial data handling* (pp. 87–104). Berlin, Germany: Springer.

Wood, C., Sullivan, B., Iliff, M., Fink, D., & Kelling, S. (2011). eBird: Engaging birders in science and conservation. *PLoS Biology*, *9*, e1001220. https://doi.org/10.1371/journal.pbio.1001220

Yuan, K., Chen, X., Gui, Z., Li, F., & Wu, H. (2019). A quad-tree-based fast and adaptive Kernel Density Estimation algorithm for heat-map generation. *International Journal of Geographical Information Science*, *33*(12), 2455–2476. https://doi.org/10.1080/13658816.2018.1555831

Zhang, G. (2020). Spatial and temporal patterns in volunteer data contribution activities: A case study of eBird. *ISPRS International Journal of Geo-Information*, *9*, 597. https://doi.org/10.3390/ijgi9100597

Zhang, G., & Zhu, A.-X. (2018). The representativeness and spatial bias of volunteered geographic information: A review. *Annals of GIS*, *24*, 151–162. https://doi.org/10.1080/19475683.2018.1501607

Zhang, G., & Zhu, A.-X. (2019). A representativeness directed approach to spatial bias mitigation in VGI for predictive mapping. *International Journal of Geographical Information Science*, *33*, 1873–1893. https://doi.org/10.1080/13658816.2019.1615071

Zhang, G., Zhu, A.-X., He, Y., Huang, Z., Ren, G., & Xiao, W. (2020). Integrating multi-source data for wildlife habitat mapping: A case study of the black-and-white snub-nosed monkey (*Rhinopithecus bieti*) in Yunnan, China. *Ecological Indicators*, *118*, 106735. https://doi.org/10.1016/j.ecolind.2020.106735

Zhang, G., Zhu, A.-X., & Huang, Q. (2017). A GPU-accelerated adaptive kernel density estimation approach for efficient point pattern analysis on spatial big data. *International Journal of Geographical Information Science*, *31*, 2068–2097. https://doi.org/10.1080/13658816.2017.1324975

Zhang, G., Zhu, A.-X., Liu, J., Guo, S., & Zhu, Y. (2021). PyCLiPSM: Harnessing heterogeneous computing resources on CPUs and GPUs for accelerated digital soil mapping. *Transactions in GIS*, *25*, 1396–1418. https://doi.org/10.1111/tgis.12730

Zhang, G., Zhu, A.-X., Windels, S. K., & Qin, C.-Z. (2018). Modelling species habitat suitability from presence-only data using kernel density estimation. *Ecological Indicators*, *93*, 387–396. https://doi.org/10.1016/j.ecolind.2018.04.002

Zhang, J. (2010). Towards personal high-performance geospatial computing (HPC-G): Perspectives and a case study. In *Proceedings of the ACM SIGSPATIAL International Workshop on High Performance and Distributed Geographic Information Systems*, San Jose, CA (pp. 3–10). New York, NY: ACM.

Zhu, A.-X., Zhang, G., Wang, W., Xiao, W., Huang, Z.-P., Dunzhu, G.-S., … Yang, S. (2015). A citizen data-based approach to predictive mapping of spatial variation of natural phenomena. *International Journal of Geographical Information Science*, *29*, 1864–1886. https://doi.org/10.1080/13658816.2015.1058387

**How to cite this article:** Zhang, G. (2022). PyCLKDE: A big data-enabled high-performance computational framework for species habitat suitability modeling and mapping. *Transactions in GIS*, 00, 1–21. https://doi.org/10.1111/tgis.12901