A MULTIDIMENSIONAL ARRAY DATABASE ENGINE FOR GRIDDED CLIMATE DATA AND A PRECIPITATION DOWNSCALING STUDY

by

Mengchao Xu
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Earth Systems and Geoinformation Sciences

Committee:

_____	Dr. Chaowei Yang, Dissertation Director
_____	Dr. Ruixin Yang, Committee Member
_____	Dr. Donglian Sun, Committee Member
_____	Dr. Liang Zhao, Committee Member
_____	Dr. Dieter Pfoser, Department Chairperson
_____	Dr. Donna M. Fox, Associate Dean, Office of Student Affairs & Special Programs, College of Science
_____	Dr. Ali Andalibi, Interim Dean, College of Science
Date: _____	Spring Semester 2020 George Mason University Fairfax, VA

A Multidimensional Array Database Engine for Gridded Climate Data and a Precipitation Downscaling Study

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

by

Mengchao Xu
Master of Sciences
University of Redlands, 2014
Bachelor of Arts
Saint John's University, 2013

Director: Chaowei Yang, Professor
Department of Geography and Geoinformation Science

Spring Semester 2020
George Mason University
Fairfax, VA

DEDICATION

This dissertation is dedicated to my wife - Ying Ju, my son - Adam Xu, and my parents – Dibao Xu, Jianping Tao.

ACKNOWLEDGEMENTS

I would like to thank all people who have helped me during my PhD studies. Firstly, I want to express my sincere gratitude to my advisor and committee chair, Dr. Chaowei Yang. I would also like to thank my committee members, Dr. Ruixin Yang, Dr. Donglian Sun, and Dr. Liang Zhao. Finally, I would like to say special thank you to my colleagues at the NSF Spatioemporal Innovation Center, without whom I would not have been able to complete this dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

Analogue Method ..... AM
Array Manipulation Language ..... AML
Array Query Language ..... AQL
Binary Association Table ..... BAT
Binary Large Objects ..... BLOBs
Central Processing Unit ..... CPU
Climate Data Online ..... CDO
Comma-Separated Values ..... CSV
Common Data Format ..... CDF
Create, Read, Update, and Delete ..... CRUD
Database Management System ..... DBMS
District of Columbia ..... D.C.
Dynamic Time Warping ..... DTW
Earth Observing System Data and Information System ..... EOSDIS
Earth Science Data Systems ..... ESDS
Empirical-Statistical Downscaling ..... ESD
Euclidean Distance ..... ED
Extreme Value Theory ..... EVT
File Transfer Protocol ..... FTP
GEOS-5 Atmospheric General Circulation Model ..... AGCM
Geospatial JavaScript Object Notation ..... GeoJSON
Gigabyte ..... GB
Gigahertz ..... Ghz
Global Climate Model/ General Circulation Model ..... GCM
Global Precipitation Measurement ..... GPM
Goddard Earth Observing System Version 5 ..... GEOS-5
GRIdded Binary ..... GRIB
Hadoop Distributed File System ..... HDFS
Hard Disk Drive ..... HDD
Hardware ..... H/W
Hierarchical Data Format ..... HDF
Information Retrieval ..... IR
Input/output ..... I/O
Integrated Multi-satellitE Retrievals for GPM ..... IMERG
Intergovernmental Panel on Climate Change ..... IPCC
Inverse Distance Weighting ..... IDW

Kilobyte ..... KB
Lightning Memory-Mapped Database ..... LMDB
Limited-Area climate Model ..... LAM
Mean Squared Error ..... MSE
Megabyte ..... MB
Model for Generating Daily Weather Variables ..... WGEN
Model Output Statistics ..... MOS
Moderate Resolution Imaging Spectroradiometer ..... MODIS
Modern Era-Retrospective Analysis for Research and Applications ..... MERRA
Modern-Era Retrospective Analysis for Research and Applications Version 2 .. MERRA-2
Multidimensional Discrete Data ..... MDD
Multidimensional Online Analytical Processing ..... MOLAP
National Aeronautics and Space Administration ..... NASA
National Center for Atmospheric Research ..... NCAR
National Climatic Data Center ..... NCDC
National Oceanic and Atmospheric Administration ..... NOAA
Network Common Data Form ..... NetCDF
Not a Number ..... NaN
Not only SQL ..... NoSQL
Online Analytical Processing ..... OLAP
Online Transactional Processing ..... OLTP
Optimal Predictor Selection ..... OPS
Perfect Prognosis ..... PP
Principal Component Analysis ..... PCA
Rainfall Filtered Autoregressive Model ..... RainFARM
Random Forests ..... RF
Regional Climate Model system ..... RegCM
Regional Climate Model ..... RCM
Relational Array Mapping ..... RAM
Relational Database Management System ..... RDBMS
Revolutions Per Minute ..... RPM
Root Mean Square Error ..... RMSE
Single Image Super-Resolution ..... SISR
Solid State Drive ..... SSD
Sparse Relational Array Mapping ..... SRAM
Spatial Data Transfer Standard ..... SDTS
Structured Query Language ..... SQL
Super Resolution Convolutional Neural Networks ..... SRCNN
Super-Resolution ..... SR
Support Vector Machine ..... SVM
United States ..... U.S.
University Corporation for Atmospheric Research ..... UCAR
Weather Research and Forecasting ..... WRF

ABSTRACT

A MULTIDIMENSIONAL ARRAY DATABASE ENGINE FOR GRIDDED CLIMATE DATA AND A PRECIPITATION DOWNSCALING STUDY

Mengchao Xu, Ph.D.

George Mason University, 2020

Dissertation Director: Dr. Chaowei Yang

Global Climate Models (GCMs) are essential tools to simulate future climate indicators and are widely used in global climate change studies. The outputs of GCMs are one of the largest sources of multidimensional gridded climate data. Data repositories are pervasive components for storing such a large amount of climate data in a big data fashion for climate studies. However, efficiently managing and querying multidimensional gridded climate data are still beyond the capabilities of most databases. The mismatch between the array data model and relational data model limited the performance to query multidimensional data from a traditional database when data volume hits a cap. Even a trivial data retrieval on large amount of multidimensional datasets in a relational database is time-consuming and requires enormous storage space. Given the scientific interests and application demands on time-sensitive spatiotemporal data query and analysis, there is an urgent need for efficient data storage and fast data

retrieval solutions on large multidimensional climate datasets. To address this challenge, I introduce a method for multidimensional data storing and accessing, which includes a new hash function algorithm, a unified data storage structure, and memory-mapping technology. A prototype database engine, LotDB, was developed as an implementation of the method, which shows promising results on multidimensional gridded climate data queries compared with SciDB, MongoDB, and PostgreSQL.

Meanwhile, climate and weather indicators such as precipitation derived from GCMs and satellite observations are important for the global and local hydrological assessment. However, most popular precipitation products (with spatial resolutions greater than 10km) are too coarse for local impact studies and require "downscaling" to obtain higher resolution. Traditional precipitation downscaling methods such as statistical and dynamic downscaling require an input of additional meteorological variables and very few are applicable for downscaling hourly precipitation for higher spatial resolution. To address this challenge, I utilized dynamic dictionary learning to propose a new downscaling method, PreciPatch, by producing spatially distributed higher resolution precipitation fields with only precipitation input from GCMs at hourly temporal resolution and large geographical scope. The second part of my dissertation details downscaling case studies conducted to evaluate the performance of the proposed downscaling method (PreciPatch) with bicubic interpolation, RainFARM – a stochastic downscaling method, and DeepSD – a super-resolution convolutional neural network (SRCNN) based downscaling method. PreciPatch demonstrates better performance than other methods for simulating short-duration precipitation events in both aggregated

IMERG data downscaling study case and MERRA-2 precipitation downscaling study case.

CHAPTER 1. INTRODUCTION

Gridded climate data are fundamental to many scientific applications and services, such as meteorology, oceanography, hydrology, agriculture, water supply, and drought. These data sets range from uniformly spaced grid points along one dimension to multidimensional grids with different variables (Barrodale Computing Services Ltd. 2002). For example, if ocean temperature were recorded every hour at every ten meters in-depth and every degree in both longitude and latitude, this would result in a 4D grid with three spatial dimensions and one temporal dimension. In the geoscience community, regularly gridded climate data is one of the most important data sources for climate analyses and has been used extensively for many years (Haylock et al. 2008; Ledesma and Futter 2017). In the past decade, both the volume and variety of the gridded climate Earth data have been further expanded. The rise of "Big Data" has brought unprecedented challenges for conventional data processing techniques and methods (Qi and Xuelong 2019). The Earth Observation data is one of the main gridded data sources in climate community, according to Earth Observing System Data and Information System (EOSDIS)'s metrics of 2014, EOSDIS manages over 9 PB of data and adds 6.4 TB of data to its archives every day (Blumenfeld 2015). In 2019, the ingest rate of data into the EOSDIS archive is projected to be 32 PB and will grow to 47.7 PB by 2022 according to the estimation from Earth Science Data Systems (ESDS) Program (NASA

EARTHDATA 2019). Such a significant increase in gridded data that occurred in the past decade marked a change in the workflow of researchers and developers (Baumann et al. 2019).

1.1 Gridded Climate Data Storing and Manipulating

In the past, gridded climate data were typically stored in simple files and then been manipulated by software programs that perform operations or analysis on these files (Barrodale Computing Services Ltd. 2002). Different standards and formats have been defined for grid data storage because of the extensive use of grids in climate modeling and analysis. Some popular formats are CDF (Common Data Format), GRIB (GRIdded Binary), HDF (Hierarchical Data Format), NetCDF (Network Common Data Form), and SDTS (Spatial Data Transfer Standard). Early approaches to handle gridded data include manual filtering and extracting after retrieving several flat files from FTP servers. Data manipulating would be either running a batch of computation processes locally or developing software running on computer clusters for single-use-cases (Baumann et al. 2018). Furthermore, these approaches are often limited to some specific data products and file formats. Changing data products in such approaches usually results in a re-write of the software or schema. Meanwhile, data volumes are exploding, and early approaches are not feasible anymore when dealing with Petabytes of data that need to be stored, filtered and processed beforehand (Baumann et al. 2018).

There is an increasing recognition that storing gridded data in modern databases and allowing users to query against them are beneficial (Barrodale Computing Services Ltd. 2002; Appel et al. 2018; Baumann et al. 2019). The major advantages of storing

grids in databases include: 1) the ability to ensure data integrity and consistency over time, 2) providing diverse users with independent and effective access to the data across applications and systems (Barrodale Computing Services Ltd. 2002), 3) flexible data retrieval or manipulation ability across a large number of datasets, 4) the ability to be rapidly searchable and selectable (Lim et al. 2009), 5) the scalability and support for big data (Madden 2012), and 6) the ability for simplifying application development and eliminating the need for users to know about how data is stored (Cudré-Mauroux et al. 2009), etc.

Grid-based data sets, raster structured data sets, array-based data sets, or data cubes are all referring to the same concept of a data structure – the array. A regular multidimensional grid with measurements is a multidimensional (n-D) array with values in the computer science context. The mainstream database technologies for multidimensional array management are relational databases, NoSQL databases, and array database technologies (Tan and Yue 2016). For DBMSs, a well-developed relational database management system (RDBMS) is based on the relational data model, and the database storage structures on lower storage levels are often trees (B+ trees). Historically, since traditional databases are not designed with the climate Earth domain in mind, nor are they built to store array data sets, RDBMS does not directly support the array data model to the same extent as sets and tables. Mapping attempts have been made to utilize the mature relational model as the backend to support array datasets, like in van Ballegooij's work (2004), and theories have been developed to support array query (Libkin, Machlin, and Wong 1996; Marathe and Salem 1997); however, with the growth

in data volume, storing, accessing, and analyzing multidimensional arrays in RDBMS became problematic and not scalable.

Meanwhile, NoSQL databases have been developed to meet the demand to store and query the increasing amount of data from various sources, in which, array data could also be mapped to different data models, like using key-value pairs to store cells of the array (Ameri et al. 2014). However, the data model mapping brings longer data pre-processing time and larger data storage volume. At the same time, as the data volume and complexity increase, there is no promising performance increase from the corresponding increase in unit computer resource consumption. Recently, the array database is drawing increasing attention because of its array data model's support on the database level, which provides a more native solution for array data storage (Brown 2010; Appel et al. 2016; Baumann et al. 2018). Tree data structures, which are usually used in relational databases' storage level on disks, have lower computational complexities for value search when comparing with an array data structure. In contrast, the array data structure offers the lowest computational complexity for data access, perfect for static data retrieval if indexes could be presented as integers. Popular array databases like Rasdaman and SciDB are widely used in many practical projects (Baumann et al. 1998; Stonebraker et al. 2013), showing promising performance gain compared to traditional methods. However, existing array databases are still suffering from the problems include: 1) low performance on non-cluster environment: the standalone version of array databases usually have limited performance comparing to their clustered setup (Hu et al. 2018), 2) long data pre-processing time: current array databases do not provide direct support for

multidimensional climate gridded data, data pre-processing is therefore necessary and cannot be avoided (Clune et al. 2015), and 3) high data expansion rate compared to raw data: due to the data pre-processing and data chunking inside databases, data volume could be expanded multiple times compared to the raw data.

Climate data management is essential and fundamental for many data related studies and applications; however, it can easily be ignored by many researchers because they usually do not have many choices for data management. Even climate data have been expanded in volume, velocity, variety, veracity, and value (Yang et al. 2017) year-by-year, the data processing and manipulating workflow have not changed much during the past decade (Baumann et al. 2019), which marked a delay in climate data management studies. Although there are many types of research utilizing distributed storage systems like HDFS and frameworks like Spark to offer solutions for big climate data storage and query, they are often focusing on implementing customized solutions and such solutions are beyond the reach of normal researches and students. Past studies on array databases have shown values and advantages in using the array data model to handle multidimensional arrays, the importance of providing database level management of gridded climate data have been increasingly recognized by climate scientists and end-users (Barrodale Computing Services Ltd. 2002; Appel et al. 2018; Baumann et al. 2019). Part of this dissertation research is inspired by such challenge and aims to provide a more cost-effective and high-performance solution for gridded climate data management and query.

1.2 Gridded Precipitation Downscaling

Broadly speaking, gridded data arises in two main areas of application: 1) modeling applications, and 2) data analytical applications (Barrodale Computing Services Ltd. 2002). The modeling applications are often involving solving a series of mathematical equations, like differential equations. Global Climate models, also known as general circulation models or GCMs (Climate.gov n.d.), are typical examples of modeling applications that produce large amounts of gridded climate data. They are fully coupled, computer-based models of the physics, chemistry, and biology of the components of Earth system (e.g. atmosphere, oceans, land surface, etc.) and their interactions with each other (Karl and Trenberth 2003). Near-surface climate variables like temperature, wind velocity, or precipitation have high impact potential on human activities (Friederichs and Hence 2007). GCMs help us obtain important scientific insights into the climate system changes (Dixon, Harris, and Knutson n.d.), and the outputs (gridded climate datasets) from GCMs played a key role in assessing the impact of large-scale climate variation and human activities. Simultaneously, such gridded data products are also feeding data analysis applications and models as further inputs for regional climate predictions and simulations. For example, the MERRA reanalysis dataset could be used as the initial and boundary forcing conditions for the WRF model in wind simulation and wind energy estimation (Carvalho et al. 2014).

The resolution of the gridded datasets, or the grid resolution, is a critical property for gridded data. Instead of calculating the climate variable traits over the whole surface of the Earth, which is beyond the reach of even the most powerful supercomputers, the

climate models use grids of "cells" to establish the locations of the "virtual weather stations" (Scied.ucar.edu. 2011). A GCM usually have grid cells with a size of about 100 km to 200 km (1° to 2.5°) on both spatial sides (Walton et al. 2015), and the "virtual weather stations" are located at the corners of the grid cells (Scied.ucar.edu. 2011; Sobhani et al. 2019). Low-resolution products or coarse-resolution datasets are often enough for understanding the changes in global patterns of climate variables like temperature; yet, assessing regional and local impacts would require higher resolution or finer resolution products (Chaney et al. 2014). However, GCM's resolutions are often too coarse to resolve important topographical features, and not sufficient to estimate and predict mesoscale processes that govern local climate changes (Giorgi and Mearns 1991; Arritt and Rummukainen 2011; Walton et al. 2015). Hence, higher-resolution scenarios of the most relevant meteorological variables are required for reliably assessing the hydrological impacts of climate change (Maraun et al. 2010). Obtaining higher resolution gridded climate data will result in better prediction of local climate change and be beneficial in regional climate estimation and analysis (Nashwan, Shahid, and Chung 2019).

Downscaling techniques were developed primarily to infer high-resolution information from low-resolution variables (Wilby et al. 1998). Downscaling is an ill-posed problem. Operationally speaking, it is a disaggregation problem, in which any number of small-scale weather sequences can be associated with a given set of large-scale values (Wilks 2010). In remote sensing, downscaling involves a process of decreasing pixel size of input imagery (Atkinson 2013). However, increasing resolution

without considering the spatial variations and local patterns of a climate variable often results in a highly biased and failed attempt. Spatial interpolations could be treated as straightforward methods to increase the resolution of a dataset, which uses "neighboring" points to predict values between known points. By implementing grids as pixels, they could be applied to downscale gridded data. However, climate events like precipitation are not continuous in space. Common interpolation methods include: 1) vector-based methods: IDW, Kriging, Spline, Nearest Neighbor, and 2) raster-based methods: Nearest, Bicubic, etc. They are not working in precipitation downscaling and often not applicable to gridded datasets.

The main technologies of climate downscaling include dynamical downscaling and statistical downscaling (also referred to as "empirical-statistical downscaling"), in which dynamical downscaling requires global climate models to support local conditions, and statistical downscaling requires local observations. Dynamical downscaling is often referred to as the process of nesting Regional Climate Models (RCMs) or direct configuration of GCMs to produce higher resolution outputs (could be highly biased). For example, the Regional Climate Model system (RegCM) was constructed at NCAR (National Center for Atmospheric Research). It has undergone continuous updates to improve its performance since first appeared in 1989 (Benestad 2016). Dynamical downscaling approaches could produce finer resolution gridded products and therefore provide much more detailed information; however, they take much more computing time to run. Generally, about ten times of the computing power will be needed if the resolution of a model is increasing by a factor of two, or the model will take ten times longer to run

under the same hardware (Scied.ucar.edu. 2011). Furthermore, RCMs are usually only available for a few areas and apply to the climate variables that are well simulated (e.g. temperature). However, precipitation is not well simulated and difficult to model (Vandal 2018). Dynamical downscaling is expensive considering its consumption in time and computer resources, so, statistical downscaling methods have been developed as alternatives. Statistical downscaling methods try to make use of the information we have on the dependency between large and local scales and then correlated with observation data (Benestad 2016). Widely used methods include: 1) Perfect Prognosis (PP) (Klein, Lewis, and Enger 1959), 2) Model Output Statistics (MOS) (Klein 1974; Bermowitz 1975), and 3) weather generator (Wilby and Wigley 1997).

For hydrological impact studies, precipitation and temperature are the most relevant meteorological variables (Maraun 2010; Xu 1999; Bronstert 2007). Long-term precipitation prediction is significant for planning flood responses, as well as strategic planning for agriculture, water resources, and other water-related hazards. Meanwhile, shorter-term predictions have more immediate applications, such as hydrological extreme prediction and management (Rau et al. 2019), and crop yields prediction for a specific area (Maraun 2010). For precipitation, it is considerably more difficult to model than temperature mostly because it is not continuous spatiotemporally. Specifically, precipitation downscaling is challenging from the following aspects:

1) The basic assumption of statistical downscaling methods is not verifiable. Specifically, it assumes the statistical relationships developed for the present day climate also held under the different forcing conditions of possible future climates

(IPCC n.d.). It is an issue not only for precipitation but also for all climate variables.

2) The data used in the statistical downscaling method may not be readily available in every region, higher resolution observation data with promising qualities are often limited to certain areas (Jarosch, Anslow, and Clarke 2012).

3) Precipitation events are not well simulated because of the high spatial and temporal variability and its nonlinear nature (Kundzewicz et al. 2007; Maraun 2010; Vrac and Naveau 2007; Vandal 2018). Precipitation products vary in spatial distribution patterns and no ground truth could be easily obtained. Our best approach is to use rain gauge, which is an indirect measurement of precipitation and could be highly biased.

4) Empirically based techniques cannot account for possible systematic changes in regional forcing conditions (Vrac and Naveau 2007)

5) A systematic assessment of the uncertainty of downscaling methods and comparison with other techniques is difficult and may need to be carried out on a case-by-case basis (IPCC n.d.).

6) In principle, downscaling can provide three types of output: point scale, areal average, or spatially distributed precipitation fields. To provide a local-scale spatially distributed precipitation field with the high temporal resolution is a challenging task (Maraun 2010). Major downscaling approaches only focus on extreme weather conditions or temporally coarse-resolution products, mostly

daily averaged precipitation on a single site (i.e. point scale) (Wilby 2004). Very few studies are generated upon hourly precipitation products.

Predictions of future precipitation changes are highly uncertain and our ability to model the local precipitation ranging from large-scale to microscale is limited (Lenderink and Van Meijgaard 2008). High temporal resolution precipitation products can provide more information than daily averages. For example, changes in hourly precipitation extremes under greenhouse warming has been found to increase twice as fast with rising temperatures compared with daily values when daily temperature mean exceeded 12 degrees Celsius, and this relationship often comes with short-duration extreme events, which can have serious impact, such as local flooding, erosion and water damage (Lenderink and Van Meijgaard 2008). However, current studies in precipitation downscaling can hardly provde a method to simulate short-duration precipitation events at the local-scale. Meanwhile, extreme precipitation events like heavy rainfall and rainstorm can cause significant societal impacts, including flash flooding, crop destruction, loss of life and infrastructure damage. To mitigate potential consequences, an understanding is needed of how rainfall will impact a region, not just a point location. Modeling of precipitation field is required in such studies (Saunders et al. 2017). Offering precipitation fields as downscaled results has been considered as a challenging test (Maraun 2010) due to many reasons, e.g. precipitation is not well simulated by either large-scale GCMs or small-scale RCMs. The precipitation downscaling part of this dissertation research was motivated by such challenges, the studies that conducted in this

dissertation aim to provide a solution for downscaling short-duration future precipitation events as precipitation fields.

1.3 Objectives and Contributions

Using new technologies to enhance big Earth data discovery, storage, and retrieval is attractive and demanded not only in the climate Earth community but also in all science domains and data-related industries (Yang et al. 2017). Meanwhile, utilizing modern technologies like machine learning to aid climate studies such as precipitation downscaling is growing in popularity.

1.3.1 Objectives

There are two main research focuses of this dissertation research: 1) investigate multidimensional gridded climate data store and query in databases, and 2) investigate gridded precipitation downscaling methods. Specifically, the research objectives are:

1) Assessing current databases for handling gridded climate datasets and proposing new methods or algorithms to accelerate multidimensional array-based climate data queries in databases. Popular databases will be tested and evaluated regarding query performance and data storage volume.

2) Investigating precipitation downscaling methods and proposing precipitation downscaling methods that could meet end user's needs for downscaling precipitation products, for example, MERRA-2 precipitation variable. As an operational request, the downscaling algorithm should be able to derive the small-scale statistical properties of rainfall from a precipitation field defined on larger

scales (Rebora et al. 2006). The method should be data independent when producing downscaled results from the input dataset

The first part of this dissertation is motivated by the challenge of finding cost-efficient and high-performance solutions for handling large multidimensional gridded datasets. While the second part is motivated by the challenge of offering high quality downscaled results for gridded precipitation products. The primary goal of this combined research is to develop methods and algorithms that could assist researchers to manipulate gridded climate datasets efficiently and provide new solutions for downscaling gridded climate datasets (e.g. gridded precipitation).

1.3.2 Contributions

The main contributions of this dissertation include:

1) An N-Dimensional hash function for a fast query on array-based climate data and a database engine as a quick implementation. The method could improve the general performance of data analysis by reducing the data storage cost and access latency. Since a large portion of climate data is gridded data, this research provides additional options for seeking cost-efficient and high-performance solutions for handling gridded data.

2) A precipitation downscaling method and associated algorithms that could simulate precipitation fields at a local scale and at the same time be consistent with large-scale information. The method does not require additional data as predictors to produce downscaled results, and it could be used for downscaling future estimations from GCMs. The downscaled results have potential to aid

climate models like WRF in the form of providing higher resolution inputs. The downscaled results could also be used to force mountain glacier models for local impact studies, where the complete absence of climate monitoring activities within the regions of interest presents a data challenge (Jarosch, Anslow, and Clarke 2012).

1.4 Dissertation Overview

The rest of this dissertation is organized as follows. In Chapter 2, the studies of storing gridded climate datasets or arrays in databases are reviewed. An n-dimensional hash function is introduced as a cost-efficient method to retrieve data from a unified data storage structure. A database engine prototype called LotDB is developed as the implementation. Query test results and performance evaluation are also included in Chapter 2. Chapter 3 focuses on the study of precipitation downscaling. The related works regarding precipitation downscaling are investigated and a new precipitation downscaling method called PreciPatch is proposed based on dictionary learning. Two case studies are generated to evaluate the performance of the downscaling method. Chapter 4 concludes this dissertation with summaries for both gridded climate data management study and precipitation downscaling study and an outlook for future works.

CHAPTER 2. A MULTIDIMENSIONAL ARRAY DATABASE ENGINE FOR GRIDDED CLIMATE DATA

2.1 Introduction

In climate community, with the growth of both data volume and complexity of spatiotemporal gridded data, the importance of efficient data storage, fast access, and analysis are well recognized by both data producers and data users. Gridded climate data management systems are the foundation of many climate studies and online services. Rapidly searching and querying abilities are affecting the speed of science investigations in climate-related researches (e.g. precipitation downscaling study). As the nature of many natural phenomena, climate can be modeled as array data sets with some dimensionality and known lengths (Baumann 1999). The studies of storing and manipulating multidimensional gridded data in databases can be generalized to: 1) array data model mapping in relational databases or NoSQL databases, and 2) array database development. Meanwhile, recent studies that using big data frameworks like Hadoop systems to store and retrieve multi-dimensional climate data sets have excellent performance results on data analysis queries (Li et al. 2017; Cuzzocrea, Song, and Davis 2011; Song 2015). However, most of the solutions were developed as customized applications for specific data formats or data sets and could not be applied to arbitrary multidimensional arrays. The related works in this section focus on the database solutions for multidimensional array storage and retrieval, and efforts done by researchers to

enhance the array data query. The output of this research could be used to boost climate data related studies like precipitation downscaling (Chapter 3).

2.2 Literature Review

2.2.1 Relational Database

The data model in relational databases are tables, most of the relational database systems do not support storing multidimensional arrays natively. Building an array model on top of existing relational databases is difficult. A natural way to store arrays in tables is to split arrays cell into rows, which is straightforward for 1D/2D datasets, however, problematic for higher-dimensional data. Operationally speaking, array orientated analysis is often done on platforms like MATLAB, which is limited by memory capacity and may have an error-prone developing process. To close this gap, in 2004, van Ballegooij introduced an array data structure in a relational database environment, developed a multidimensional array DBMS called RAM (Relational Array Mapping) which maps array onto a traditional relational schema. The multiple index columns in a multidimensional array were compressed into a single column in their approach. Data retrieval process was based on existing database system with bulk array processing. RAM aims to utilize the advantages of the mature database system and storage structure, adapting to an array model to fit in the existing DBMS to ease the array storage and analysis process. RAM also designed a query language for an array data structure in the relational model which composed of two methods: data extraction from array storage, and construction of arrays. They provided an intellectual framework and ease the use of map arrays to relational models in a distributed application as the implementation (van

Ballegooij et al. 2005). However, mapping the array model to relational could only be treated as a compromised solution rather than the ideal solution.

In 2005, the prototype of MonetDB was introduced as a main memory database system that uses a column-at-a time execution model for the data warehouse. Although MonetDB is not a native array database but a full-fledged relational DBMS (Idreos 2012), it provided useful thoughts and ideas for processing array models. It is a column-oriented database and each column, or BAT (Binary Association Table), in the database is implemented as a C-array on storage level. (Boncz, Zukowski, and Nes 2005) MonetDB has focused on optimizing the major components of traditional database architecture to make better use of modern hardware in database applications that support massive data volumes (Boncz, Kersten, and Manegold 2008). MonetDB is a column-based storage engine that favors analysis queries by better-using CPU caches. The analysis is focusing on highly CPU-efficient execution. In 2008, Cornacchia et al. also introduced an example of using a matrix framework with a Sparse Relational Array Mapping (SRAM) by Information Retrieval (IR) researchers, they used MonetDB/X100 as the relational backend, which provided fast response and good precision. The matrix framework is based on the array abstraction, and by mapping them onto the relational model and develop array queries, MonetDB allows them to optimize the performance and develop a high-performance IR application. In general, relational databases are designed and optimized for Online Transactional Processing (OLTP) operations, which are not ideal for Online Analytical Processing (OLAP) analysis.

2.2.2 NoSQL Database

Main NoSQL (Not Only SQL) database categories are key-value, document, column, and graph. All of them can store large volumes of semi-structured or unstructured data with high Read/Write throughput (Amirian, Basiri, and Winstanley 2014). Lakshman and Malik (2010) presented a key-value based NoSQL database (Cassandra) approach to handle very large amounts of structured data, which has been found performance gain compared to the relational database system when evaluated with persistency and I/O operations on genomic data (Aniceto et al. 2014). However, key-value databases (e.g., Cassandra) provide limited functionality beyond key-value storage and have no support for relationships. Other representative key-value systems include Memcached, Aerospike, Redis, Riak, Kyoto Cabinet, Amazon DynamoDB, and CouchDB. A document-based NoSQL database such as MongoDB is used for managing geospatial data more effectively than the key-value database because geospatial data inside the document database can be retrieved using more flexible queries than key-value based database, and many document databases support geospatial data natively, for example, through GeoJSON format. Also, proximity queries can be efficiently implemented using the document database (Amirian, Basiri, and Winstanley 2014). Using MongoDB to store and access climate satellite data, Ameri et al. (2014) compared the results with a SQL database (MySQL). The performance was improved up to a factor of 46 using 11 threads on a 12-core system. However, relationships and joins are not supported as in relational databases, and such systems lack native support for multidimensional data. Other document databases include Couchbase, RavenDB, and

FatDB. For the column-based NoSQL databases, the queries are limited to keys and, in many cases, do not have a way to query by column or value, requiring a mapping layer to handle highly connected geospatial data, which is not efficient. Han and Stroulia (2013) demonstrated a data model of stored location data based on the column database (HBase). Although it does not have clear advantages regarding query response time, it scales well and supports efficient performance for range and k-nearest neighbor queries. Other systems in this category include Google BigTable, Cloudata, and Cassandra (also a key-value database).

2.2.3 Array Database

Array databases are sometimes subsumed under the NoSQL category. More recent views have treated multidimensional arrays as Multidimensional OLAP (MOLAP) data cubes in the context of Data warehouses and OLAP systems (Baumann et al. 2018), which are business intelligence technologies that allow decision-makers to analyze huge volumes of data on the fly according to the multidimensional data model (d'Orazio and Bimonte 2010). The main difference is the OLAP data cubes are rather sparse (Naydenova and Kaloyanova 2010) whereas gridded climate data tend to be rather dense (i.e. few grids hold non-null value). Research on array DBMS usually includes two parts, the storage of multidimensional array and query language (Baumann et al. 1999). The multidimensional database has a long history as the statistical database has been studied. While OLAP often focuses on business array data, scientific studies on computing and imaging have been developed formal concepts on array data manipulation, like AQL (Libkin, Machlin, and Wong 1996) and the array manipulation language AML (Marathe

and Salem 1997), but not been implemented and evaluated as real-life applications before Rasdaman (Baumann 1999). Also, many formal models are not designed for array operations; instead, they are for the goal of mapping arrays to relations tuples that do not provide a general solution. A declarative query language suitable for multiple dimensions was firstly introduced by Libkin, Machlin, and Wong (1996), Rasdaman introduced its algebraic framework in 1999 for expressing cross-dimensional queries, for which consists of only three operations: an array constructor, a generalized aggregation, and a multidimensional sorter (Baumann 1999). Although the structure is not complicated, it is sufficient to cover a wide range of imaging, statistical, and OLAP operations. On the data model level (Baumann et al. 1999), Rasdaman integrates an array model into the existing overall model, which is based on a formal algebraic framework and developed with declarative multidimensional discrete data (MDD). MDD is the data storage unit in Rasdaman. On storage level, binary large objects (BLOBs) are the smallest units of data access (Reiner et al. 2002), and the size stored in Rasdaman is from 32KB to 640 KB. MDD object is divided into arbitrary tiles or subarrays and combined with a spatial index to access the tile object used by a query (Baumann et al. 1998). On a practical level, Rasdaman introduced array as a new attribute type in Postgres, utilizing the existing relational model to adapt multidimensional arrays. The metadata is also stored in the relational model.

Two years later, SciDB was first debuted in 2009 by Cudré-Mauroux et al. It provides a native array data structure on storage level and supports clustering (Brown 2010). The system architecture is a shared-nothing design. SciDB itself is a distributed

storage manager, and data in the array is expendable but not directly updateable (no-overwrite); the updated values form a new array in SciDB. The arrays are vertically partitioned; the attributes are split in a single logical array and values are handled separately. The motivation is similar to column-store systems, like MonetDB, vertical partitioning reduces I/O cost since users often focus on a particular query on a subset of attributes in a logical array. The arrays are decomposed into potentially overlapping chunks. Different from Rasdaman (which store chunking information in the relational model), the metadata, location information of chunks are also stored within the logical array. The size of a chunk could be an order of 64 MB, which is much larger than Rasdaman and MonetDB. Since the chunks are potentially overlapped, this characteristic increases the total data volume stored in the database (Brown 2010). SciDB also provided a new way of dealing with "not valid" cells, in which no space is allocated in the array data chunk for such cells (Stonebraker et al. 2013). At the same time, SciDB optimizes CPU performance by performing vector processing, which was from the contribution of MonetDB. The version 1 of SciDB was released in 2010, the same year, a science benchmark paper (SS-DB) was published which included a comparison between SciDB and MySQL (Cudré-Mauroux et al. 2010), in which SciDB presented a high-performance potential. In 2012, Planthaber, Stonebraker, and Frew provided an integrated system of using SciDB to analyze MODIS level 1B data. They pre-processed MODIS data from HDF format to CSV, then CSV to SciDB DLF files as a system. The SciDB engine showed a promising performance and being the most powerful competitor of Rasdaman. Three years after SciDB was first announced, a SQL-based query language called SciQL

was introduced in the relational model for scientific applications with both tables and arrays as first-class citizens (Kersten et al. 2011; Zhang et al. 2011). In nature, it is an array extension to MonetDB and makes MonetDB effectively function as an array database. However, it still involves data model mapping and is not a fully operational solution.

In 2013, Rasdaman added the in-situ support for tiff files and NetCDF files (Baumann, Dumitru, and Merticariu 2013). The same year, Dr.Stonebraker, one of the main starters of SciDB (who is also the author of Postgres, VoltDB, etc.) pointed out why Hadoop is not a good solution for scientific data processing in the following three problems: 1) Problems that look like “grep”, 2) Problems that look like queries, 3) Problems that look like data analytics. The same year, Dr. Stonebraker et al. (2013) also pointed out that the SciDB research directions were in the areas of elasticity, query processing, and visualization. He also presented an example using SciDB for MODIS processing pipeline. Additional researches have been done in the recent years to customize SciDB. Soroush and Balazinska (2013) developed TimeArr as a new storage manager for SciDB, in which speeds up the process of creating or selecting a specific version of arrays, allowing researchers to explore how time changes affect the cell values in a specific array section. Liu et al. (2014) proposed a window aggregation method upon SciDB arrays to speed up the window aggregation query. One remarkable research was done by them; they extended SciDB to use GPUs and improved its query performance by 1.5X to 11X. Simultaneously, they found out that the array partitioning, load imbalance, and CPU-GPU hybrid execution were the bottlenecks that limited their steps. Unlike

CPU, which benefits from small chunk size, larger chunking size increases the performance due to lower scheduling and data transfer overheads.

To summarize, without the support of standard databases, scientists tend to employ customized programs or file-based implementations serving and analyzing array data (Baumann, Dumitru, and Merticariu 2013). Array DBMSs like Rasdaman (Baumann 1998), SciDB (Cudré-Mauroux et al. 2009), and SciQL (Kersten et al. 2011) fill this demand by extending the supported data structure in the database with unlimited-size multidimensional arrays. However, mapping the array data model to the relational model is a performance sacrifice, limited the data access ability. Also, data tiling and chunking are good for clustering and mapping to the non-array data structure. However, it causes the problem of data volume increase and decreases the I/O performance on secondary storage by using random access instead of sequential access.

2.3 Methodology

Array databases have shown the potential to be valid solutions for storing and manipulating multidimensional array data. However, current solutions fail to provide good support for multidimensional climate gridded data and face issues related to overall performance. In this section, a new data management solution is presented as the foundation to build an array database. In general, a valid data management solution for data storage and retrieval includes 1) a data storage structure and 2) a data access and update method. Specifically, the solution should have the essential functions of data manipulating: create, read, update, and delete (CRUD). To feed the needs, a unified data

storage structure is adopted for persistent storage of data on secondary-storage, and an n-dimensional hash function is proposed as the main algorithm for data manipulation.

2.3.1 A Unified Data Storage Structure

For array data structure on entity-level, the array data structure provides $O(1)$ complexity for single data retrieval; this is the fastest way computers can do when retrieving something. Traditional ways involve tiling and chunking big arrays (Baumann et al. 1999; Boncz et al. 2005; Cudré-Mauroux et al. 2009). Typically, tiling and sub-setting are necessary for array database, like in Rasdaman, the size of its smallest element is from 32KB to 640KB, as the order of default page file size (4KB). However, arrays are not suitable for fragmenting because they will increase data access and search complexity. Since disks are designed to do the sequential reading, too little or too much tilling is not good for I/O performance. In 2002, Reiner developed an R + tree for MDD tile node searching (Reiner et al. 2002); however, since the storage rectangles are duplicated, the R+ tree increases the data volume and makes construction and maintenance more expansive. In computational complexity concern, data retrieve complexity will be $O(\log_M n) + O(1)$. Ideally, a multidimensional array will provide $O(1)$ complexity when retrieving data from it. However, because of tiling and sub-setting, the complexity for search increases considerably if the tiling unit is small and the whole dataset is large. Meanwhile, when a multidimensional array is stored in a secondary storage device, data files are often not sequentially stored on the device and the cost to access different byte addresses is different. Although the computational complexity is the same for each cell in an array, the cost of finding the index and let the

disk head (in HDD) to retrieve is different. Even for SSDs, which does not have a physical head inside, random access and sequential access varies largely in speed. Since the ability to do random read/write and sequential read/write is different, storing data bytes closer to each other will have better performance for data read when they are retrieved together. Therefore, instead of chunking and tiling, putting a multidimensional array in a holonomic array form should have a performance gain for sequential data retrieval.

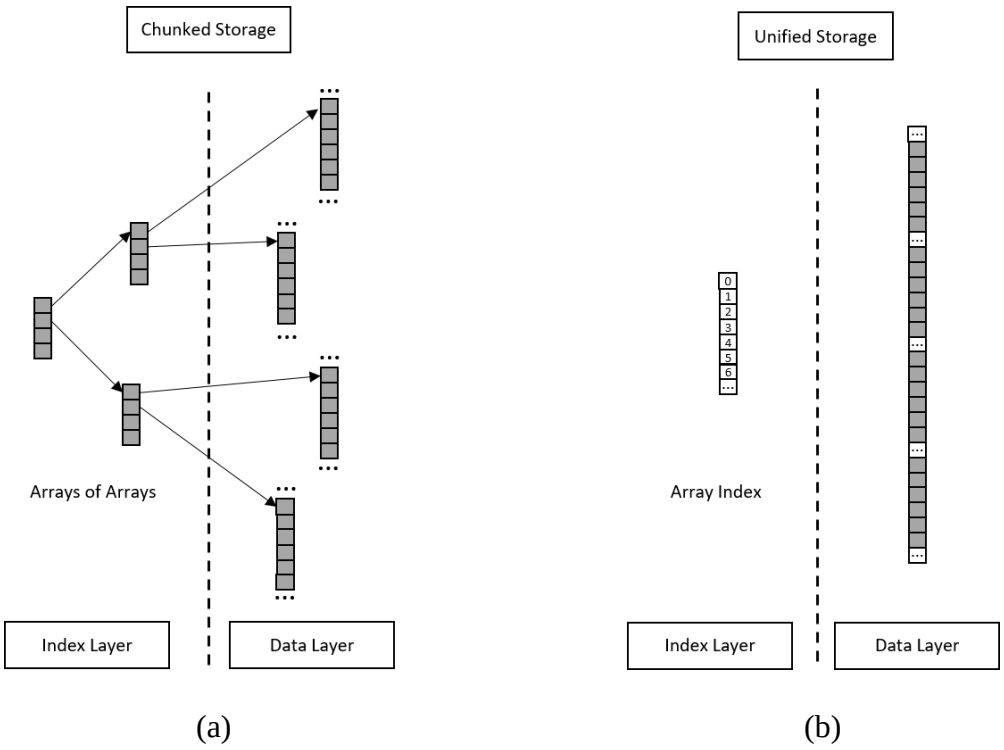


Figure 1 Chunked Storage and Unified Storage: (a) chunked storage with multilayer array indexing, (b) unified storage with no extra indexes

Figure 1 illustrates the differences in storage structure between a chunked storage structure and a unified storage structure. The chunked storage requires additional indexing for data storage and retrieval; however, a unified storage has the native array index as its storage index. Another benefit of using a unified storage is the possibility of saving storage space. Specifically, if a multidimensional array Z has N dimensions, and the size for the ith dimension is S_i, then when it stores in a unified form, it takes the B_L space in terms of bytes as expressed in the following (Equation 1):

Equation 1 Data size

$$B_L = B_d * \prod_{i=1}^{N} S_i \text{ , for } B_d = \text{data type size}$$

It is the minimum space an array will cost without any compression. For example, if a 5-D array (32-bit float) with dimensions $10 \times 5 \times 6 \times 4 \times 7$ will cost about 33KB. The HDF and NetCDF file formats use similar structures to hold a multidimensional array along with other information integrated. In practical use, it requires additional programming skills to access the data set, and a full data set load to memory space. Although a specific cell location could be calculated through its dimension sizes, there is no indexing function to map from the original multidimensional array to its unified form for subsetting.

2.3.2 An N-Dimensional Hash Function for the Unified Storage Structure

The unified storage structure provides a one-dimensional native index for each cell of it. Querying an n-dimensional array needs a hash function to map the n-dimensional index to the one-dimensional native index. In this section, I propose an n-dimensional Hash Function Algorithm to fill the demands for fast data retrieval based on

the unified array data storage structure. The detailed procedures for this algorithm are described in Table 1.

Table 1 An N-Dimensional hash function

Name: Computation of 1-D storage index from n-D query index

Definition:

(1) Z is a N dimensional array
(2) The dimensions form an ordered set as $\mathrm{D} = (\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3, \dots, \mathrm{A}_n)$.
(3) The data storing order follows the same order as D. The size of each axis forms another ordered set as $\mathrm{S} = (\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3, \dots, \mathrm{S}_n)$.
(4) Query a subset of Z is expressed as
"Find($\mathrm{A}_1\{\mathrm{Min}_1, \mathrm{Max}_1\}, \mathrm{A}_2\{\mathrm{Min}_2, \mathrm{Max}_2\}, \mathrm{A}_2\{\mathrm{Min}_2, \mathrm{Max}_2\}, \dots, \mathrm{A}_n\{\mathrm{Min}_n, \mathrm{Max}_n\}$)"

Input:

Array $\mathrm{A}_z = [\mathrm{S}_1, \mathrm{S}_2, \mathrm{S}_3, \dots, \mathrm{S}_n]$

Array $\mathrm{A}_{MinMax} = [\mathrm{Min}_1, \mathrm{Max}_1, \mathrm{Min}_2, \mathrm{Max}_2, \mathrm{Min}_3, \mathrm{Max}_3, \dots, \mathrm{Min}_n, \mathrm{Max}_n]$,
for $h \in \mathbb{Z}$: $h \in [1, n], \mathrm{S}_h \geq \mathrm{Max}_h \geq \mathrm{Min}_h \geq 1$

Array $\mathrm{A}_T = [\mathrm{s}_1, \mathrm{s}_2, \mathrm{s}_3, \dots, \mathrm{s}_n]$, for $h \in \mathbb{Z} : h \in [1, n], \mathrm{s}_h = (\mathrm{Max}_h - \mathrm{Min}_h + 1)$

Output:

$$\text{Array } \mathrm{A}_I = [\mathrm{i}_1, \mathrm{i}_2, \mathrm{i}_3, \dots, \mathrm{i}_m], \text{for m} = \sum_{h=1}^{n} (\mathrm{s}_h)$$

Procedure:

(1) **Compute** the first element of A_I: i_1

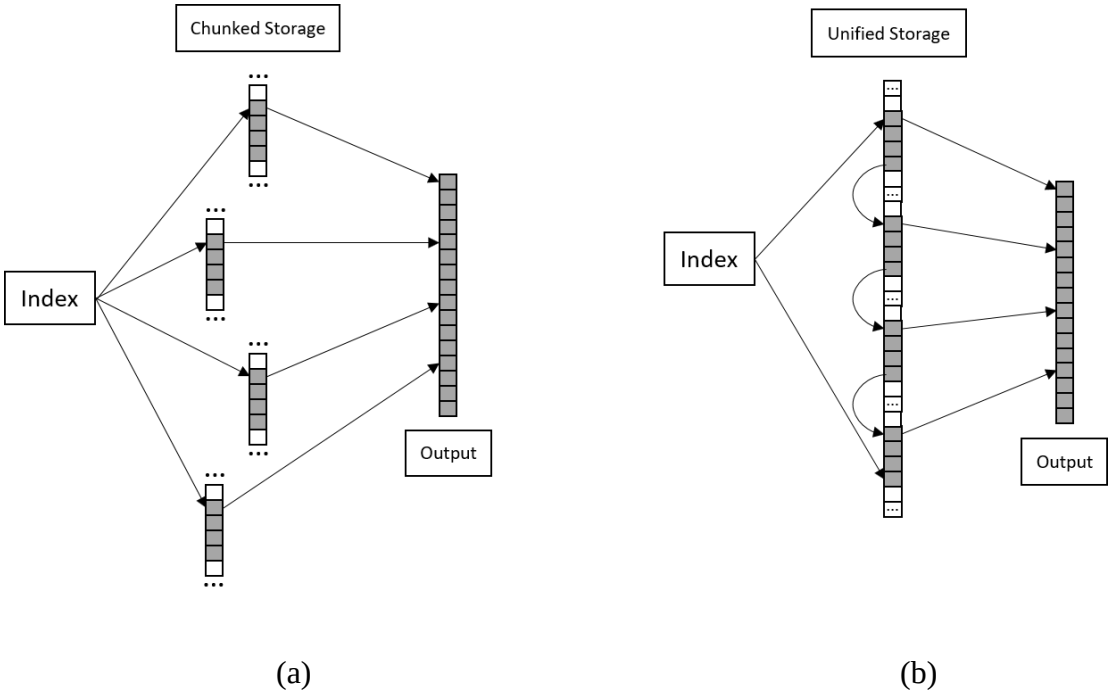
$$\mathrm{i}_1 = \sum_{h=1}^{n} (\mathrm{Min}_h * \prod_{k=h+1}^{n} (\mathrm{S}_k)), \text{ for } \mathrm{S}_{n+1} = 1$$

(2) **Compute** the last element of A_I: i_m

$$\mathrm{i}_m = \sum_{h=1}^{n} (\mathrm{Max}_h * \prod_{k=h+1}^{n} (\mathrm{S}_k)), \text{ for } \mathrm{S}_{n+1} = 1$$

(3) **If** ($\mathrm{i}_1 = \mathrm{i}_m$) **then** {
Return $\mathrm{A}_I = [\mathrm{i}_1]$ }
(4) **Else**: index $\mathrm{D}_{index} = \mathrm{i}_1$
(5) **Compute** array $\mathrm{A}_{Stepsize} = [\mathrm{g}_1, \mathrm{g}_2, \mathrm{g}_3, \dots, \mathrm{g}_{(n-1)}]$, for $p \in \mathbb{Z} : p \in [1, (n-1)]$
$\mathrm{g}_{(n-1)} = (\mathrm{S}_{(n-1)} - \mathrm{s}_{(n-1)}) + 1$
$\mathrm{g}_{(p)} = \mathrm{g}_{(p+1)} + (\mathrm{S}_{(p)} - \mathrm{s}_{(p)}) * \sum_{i=(p+1)}^{n} (\mathrm{S}_i)$
(6) **For** ($\mathrm{R}_1 = 1$ **to** $\mathrm{R}_1 = \mathrm{s}_1$) **do** {
For ($\mathrm{R}_2 = 1$ **to** $\mathrm{R}_2 = \mathrm{s}_2$) **do** {
For ($\mathrm{R}_3 = 1$ **to** $\mathrm{R}_3 = \mathrm{s}_3$) **do** {
…

For ($R_n = 1$ **to** $R_n = s_n$) **do** {
Appending D_{index} as the last element of A_I
If ($R_n \neq s_n$) **then** { $D_{index} = D_{index} + 1$ }
}
…
If ($R_3 \neq s_3$) **then** { $D_{index} = D_{index} + g_3$ }
}
If ($R_2 \neq s_2$) **then** { $D_{index} = D_{ind} + g_2$ }
}
If ($R_1 \neq s_1$) **then** { $D_{index} = D_{index} + g_1$ }
}
(7) **Return** A_I



(a) (b)

Figure 2 Index on chunked storage and unified storage, (a) current index, (b) proposed index

As a result, if the total number of data for retrieval is G, the n-dimensional hash function will hold an $O(G)$ complexity. Figure 2 shows the difference between the idea of the traditional indexing method and the proposed indexing method. In the traditional

indexing method, executing a data retrieve query forces the index to locate the initial pointers to each chunk and return with the integrated output. In the proposed indexing method, the indexing system is only used to calculate the first and last elements and a jumping step set ($A_{Stepsize}$) and data are retrieved sequentially with specific jumping steps. Typically, algorithms are designed independently from hardware (H/W), which may lead to a diminishing return of actual performance. The intention to design this algorithm is to consider the H/W limit while performing data retrieval action. Specifically, by implementing a sequential-like data retrieval action, the n-dimensional hash function algorithm will utilize the sequential read ability in disks when it is possible. It is similar to the motivation of building MonetDB to break the memory wall (Boncz et al. 2008). Also, climate gridded data queries often rely heavily on value retrieval than value search. The n-dimensional hash function and the unified storage structure are designed to maximize the value retrieval ability than value search.

2.4 Implementation

2.4.1 System Design

LotDB is a prototype array database library developed with C++ and the memory-mapping technology; it is an implementation of the n-dimensional hash function and unified data storage design. It follows a lightweight, standalone, and single-use design. LotDB acts as both a client-based database manager and a database library for applications on top, like Google's LevelDB.

2.4.2 Memory-Mapping Technology

In LotDB, data are stored in the secondary storage system, and the access to it is done by utilizing memory-mapping technology and through page files. This technology is widely used in database systems like LMDB and MongoDB. Specifically, instead of loading the whole file into memory, the file handler maps the file to virtual memory as a big array and assigns a virtual memory address to each page file without loading any actual data into the memory other than the file's metadata. When a data access call is made for a page file, it will cause a page fault and enable the read/write of the secondary storage. This way, bytes are copied to actual memory addresses directly, and no need to go through disk caches as the standard open/write will do. Also, by utilizing memory-mapping of arrays, LotDB could exceed the memory cap for accessing large data files and make it possible for LotDB accessing big arrays without tiling. Meanwhile, when integrating with the n-dimensional hash function, the array indexes could be virtually calculated with low costs and could increase data retrieve speed exponentially when compared with traditional database solutions.

2.4.3 LotDB System Architecture

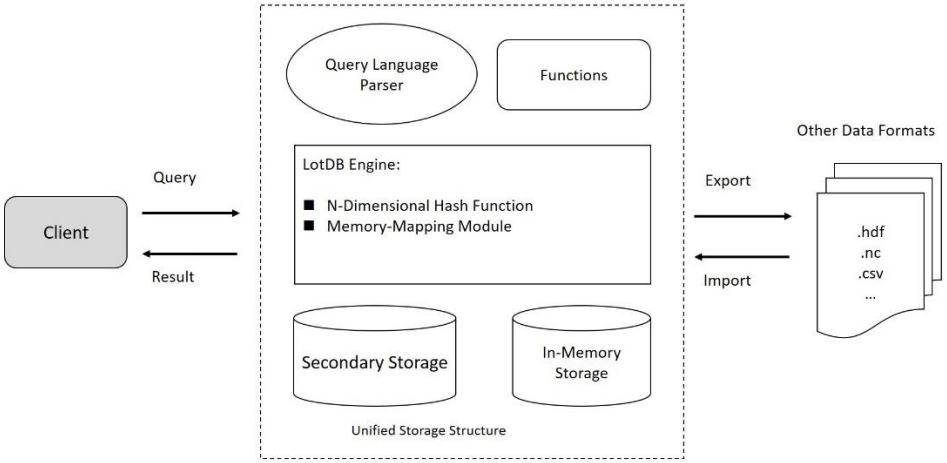


Figure 3 Architecture of LotDB

The general architecture of LotDB is shown in Figure 3. The system core is the LotDB engine, which consists of the n-dimensional hash function algorithm and memory-mapping module. Data are stored in files and separated from their metadata files; data retrieve queries are called through built-in functions, and results are stored in memory. Functions are designed to perform different calculations and services, including upload multidimensional arrays directly from their original formats like HDF or NetCDF, etc. Meanwhile, the query communications are done through a query language parser and a client. For spatiotemporal grid-based climate datasets, they are often produced and packaged without the demand of over-write. The empty cells in a grid-based climate data product do not require adjustments when storing. Such cells, or often referred to as noise, will only be eliminated when the analysis is performed and will stay with the original data. LotDB is designed to meet this characteristic of climate datasets, it will be fast to store static array data which does not change once captured. The drawback of this is that

data updating will be very expensive if it involves changes in array shape or size and the empty cells cost the same for storage as the non-empty cells. Although keeping the empty cells does waste storage space and is not ideal for sparse data, it accelerates the data retrieval by simplifying the data storage architecture on disks.

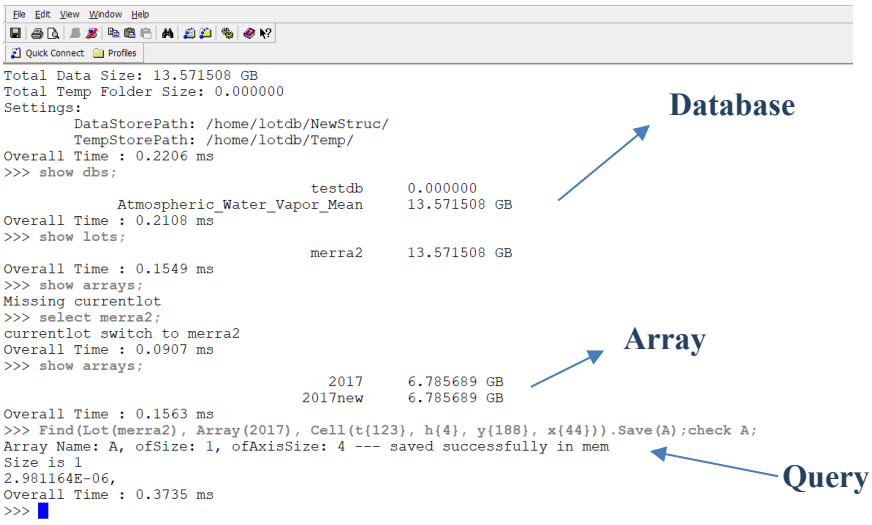


Figure 4 A glance view of LotDB

Figure 4 provides a glance view of LotDB. Arrays are stored in a database and can be manipulated through queries.

2.5 LotDB Query Tests and Performance Evaluation

In this section, the proposed array database engine LotDB was tested against three modern databases (PostgreSQL, MongoDB, and SciDB) to evaluate their performance on handling multidimensional gridded climate data.

2.5.1 Experiment Setup and Design

2.5.1.1 Introduction

LotDB is designed and developed as a quick implementation of the unified storage structure and the N-Dimensional hash function in chapter 3. The primary goal for this part of the research is to improve gridded climate data query performance for possible geoscience applications and reduce storage volume at the backhand to offer as a cost-efficient solution for climate scientists. Although LotDB is not a fully developed database system but a simple database engine for multidimensional array data storage and query, it is still worthwhile to evaluate its performance with modern databases or popular solutions that researchers are currently adapted when handling multidimensional gridded climate data.

2.5.1.2 Experiment Design and Objectives

The general design of this experiment is to store the same climate datasets into different databases or containers and use the same query contents to query against different databases installed under the same hardware environment. The performance is evaluated to conclude for databases' abilities on handline multidimensional gridded climate data.

2.5.1.2.1 Environment

All the databases are installed as standalone modes on individual servers with the same hardware configuration: Intel Xeon CPU X5660 @ 2.8Ghz×24 with 24GB RAM size and 7200 rpm HDD.

2.5.1.2.2 Databases

To compare the performance of LotDB with other databases, three popular databases are chosen with their commonly used versions; specifically, they are PostgreSQL 9.3 (Relational Database), MongoDB 4.0.4 (NoSQL Database), and SciDB 18.1 (Array Database). PostgreSQL is an open-source object-relational database management system; it has been launched for 30 years and maintained a very stable performance in different domains. MongoDB is a document-oriented NoSQL database system, which follows a schema-free design and is one of the most popular databases for modern applications. SciDB, as mentioned in the previous chapters, is a high-performance array database designed specifically for storing and querying scientific datasets. The choice of these three databases covers the most popular database solutions for handling climate gridded data. CentOS 6.6 is chosen to hold SciDB, and Ubuntu 14.04 is chosen to hold MongoDB, PostgreSQL, and LotDB. Different OS is chosen because SciDB 18.1 could not be easily configured under Ubuntu. Both CentOS and Ubuntu are utilizing common Linux kernel, newer kernel versions may carry slight improvements in performance; yet, such improvements are rarely important to database users.

2.5.1.2.3 Query Design

Different spatiotemporal queries are designed to evaluate the performance of selected databases (Table 2) for the year 2017 with a raw data size of 3.45 GB. Specifically, nine queries were designed to simulate end-users' normal operations on gridded climate data. Table 2 lists these queries in a detailed explanation and estimates

the corresponding grid points each query will retrieve from the data containers (databases).

Table 2 Spatiotemporal Queries

Index	Query Content	Number of Grid Points Retrieved
Q1	What's the precipitation in D.C. at 9:30 a.m. on August 1st, 2017?	1
Q2	What's the precipitation in D.C. from 9:30 a.m. to 21:30 p.m. for each day in June 2017?	403
Q3	What's the precipitation in D.C. from 9:30 a.m. to 21:30 p.m. for each day in 2017?	4,745
Q4	What's the precipitation in the Chesapeake Bay at 9:30 a.m. on August 1st, 2017?	72
Q5	What's the precipitation in the Chesapeake Bay from 9:30 a.m. to 21:30 p.m. for each day in June?	29,016
Q6	What's the precipitation in the Chesapeake Bay from 9:30 a.m. to 21:30 p.m. for each day in 2017?	341,640
Q7	What's the precipitation in the U.S. at 9:30 a.m. on August 1st, 2017?	4,753
Q8	What's the precipitation in the U.S. from 9:30 a.m. to 21:30 p.m. for each day in June 2017?	1,915,459
Q9	What's the precipitation in the U.S. from 9:30 a.m. to 21:30 p.m. for each day in 2017?	22,552,985

2.5.1.2.4 Data Ingest Volume Design

Different raw data sizes are chosen to evaluate the data storage consumption in different databases in addition to 3.45 GB. Specifically, they are 10 MB, 100 MB, 1 GB, and 10 GB. The number of grid points is the estimated number of array cells to be retrieved for the corresponding query

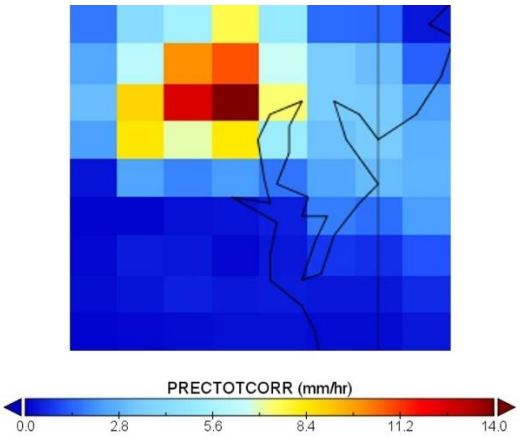
2.5.1.2.5 Validation and Evaluation

Data validation processes are executed for each run of query tests and after data importing to databases. Raw data is treated as the guideline for data accuracy, grid points retrieved from databases are compared against the raw data to check for errors or missing values. For the evaluation part, the databases are evaluated in the following aspects: 1) data uploading and pre-processing time, 2) data storage volume consumption, and 3) spatiotemporal query run-time. The query run-time refers to the elapsed real-time or wall-clock time in this experiment.

2.5.1.3 Multidimensional Datasets

The MERRA-2 dataset is collected and selected as the experiment data, which is produced and provided by the Global Modeling and Assimilation Office of NASA Goddard Space Flight Center. MERRA-2 dataset is stored in NetCDF4 format and contains about 49 variables (e.g., Surface Wind Speed, Precipitation, Surface Air Temperature, etc.). PRECTOTCORR is chosen as the variable to use in this experiment, which is the bias-corrected precipitation output from an atmospheric model. The spatiotemporal resolution of this variable is 0.625° by 0.5° with hourly reads for each day of the year. The year selected for this experiment is 2017, and the dataset is a 4D array with dimensions $365 \times 24 \times 361 \times 576$, and its data type is 32-bit float. This one-year dataset contains around 1.8 Billion grid points.

As an example, Figure 5 shows the plot view and the array view of this dataset at one time-step in the Chesapeake Bay area.



(a)

	1	2	3	4	5	6	7	8
1	0.0653923	0.153208	0.170857	0.232279	0.250947	0.189900	0.180513	0.358021
2	0.174183	0.294614	0.473785	0.338924	0.386238	0.331736	0.333774	0.608325
3	0.121719	0.394821	0.329590	0.139582	0.347185	0.789213	0.673771	1.20249
4	0.101843	0.0595987	0.221872	0.324225	0.441813	1.76210	1.54238	2.27880
5	0.288820	2.36979	1.85995	2.28825	1.69473	2.42729	2.96288	2.82040
6	2.31743	8.44231	7.00207	8.39081	5.16186	3.10535	3.45039	2.73886
7	3.01695	8.88519	12.6068	14.0213	7.34367	3.54137	3.39890	2.34232
8	2.49510	6.16779	10.0971	10.8421	6.61755	3.52077	3.05643	1.33982
9	1.71576	4.26407	5.36613	7.68356	5.00736	1.49817	1.46255	0.337529

(b)

Figure 5 MERRA-2 in Panoply: (a) grid dataset in plot view, (b) grid dataset in the array view

2.5.1.3.1 Data Preprocessing

Data preprocessing are used for databases that do not take NetCDF file as the import format. NetCDF files are converted into tables in CSV format as a data preprocessing step.

2.5.1.4 Experiment Procedure

The general procedure for all databases follows the same pattern: 1) data import or data preprocessing and then data import, 2) data validation check for stored data sets in

databases, 3) spatiotemporal data query tests and data validation for each test run, and 4) database evaluation.

2.5.2 Experiment Results and Analytics

PostgreSQL and MongoDB do not have native support for NetCDF format, and SciDB does not have a stable and fully functional plugin for NetCDF data import. Therefore, data were converted into the CSV format and then uploaded to each database. Meanwhile, LotDB is developed with data import functions to upload multidimensional data directly from the NetCDF/HDF format. To accelerate the preprocessing process, SSDs are utilized to store the raw datasets and intermediate results. Then, the preprocessed results were uploaded to servers. No additional indexes were built for all databases, data were dumped to the databases with their basic indexes. Table 3 lists the processing time and intermediate data size in CSV format, from 120 MB to 168 GB as the raw data increases from 10 MB to 10 GB.

Table 3 Preprocessing Data (from NetCDF to CSV)

Raw Data (NetCDF format)	**10 MB**	**100 MB**	**1 GB**	**3.45 GB**	**10 GB**
Intermediate Data (CSV format)	120 MB	1.4 GB	16.3 GB	52 GB	168 GB
Pre-processing Time (hours)	0.01	0.13	1.4	4	13.4

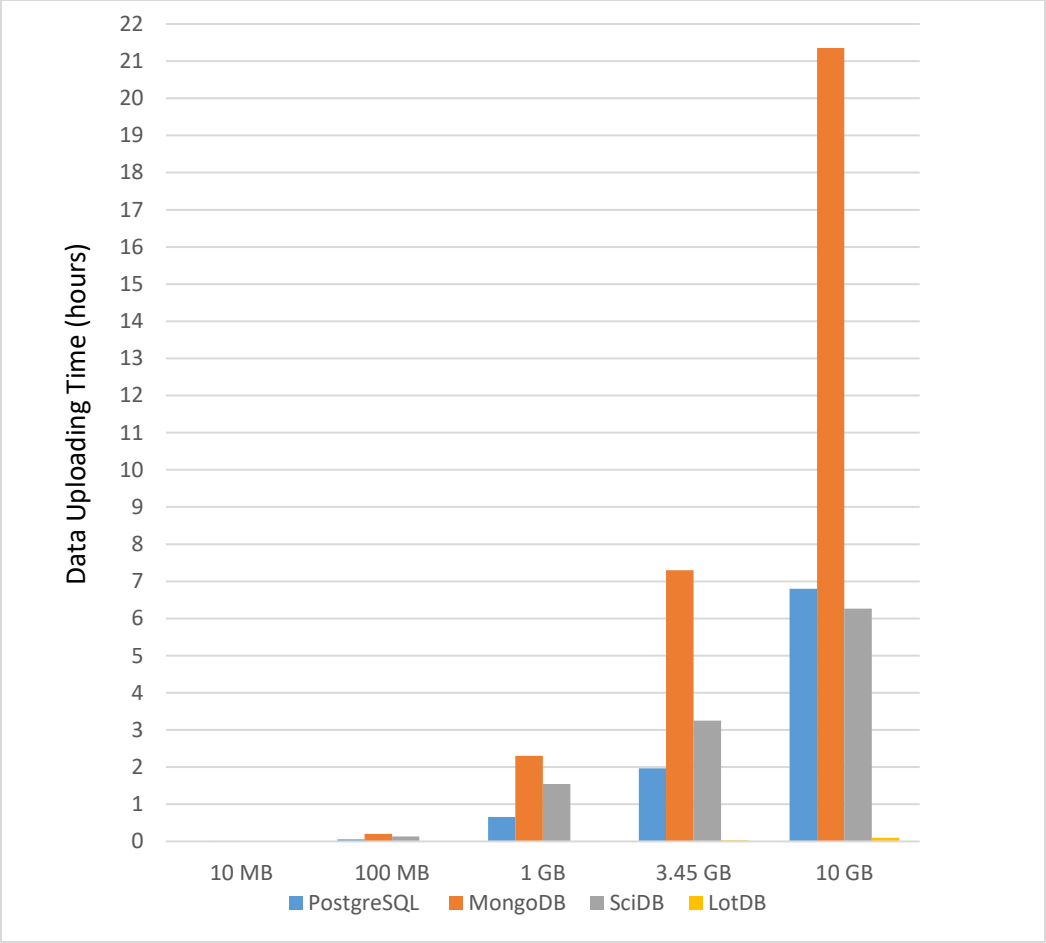


Figure 6 Data Uploading Time for PostgreSQL, MongoDB, SciDB, and LotDB

After the pre-processing, PostgreSQL, SciDB, and MongoDB all require additional time for data uploading. The detailed time variation is shown in Figure 6. The vertical axis records the data uploading time to each database from CSV files, and the horizontal axis represents different uploading cases for different raw data size. The data uploading time in this figure is independent of the pre-processing time. As the figure shows, MongoDB took the longest time for data uploading in almost all the cases. Meanwhile, LotDB used the least time amount for data uploading because of its native

support for NetCDF data files and unified data storage design. As a representative of a relational database in this experiment, PostgreSQL shows a stable increase in uploading time as raw data size increases. It has an advantage when dealing with a small number of array datasets compare to MongoDB and SciDB. SciDB is increasing in its data uploading time with a decrease in speed change, which implies a potential advantage in handling larger datasets. MongoDB has the highest rate of time complexity while LotDB has an almost linear rate for data uploading. As designed in the LotDB data import function, data are dumped directly from the raw dataset if it was stored linearly in multidimensional array data formats like NetCDF. Therefore, it is not surprising that LotDB has significant advantages in multidimensional data uploading.

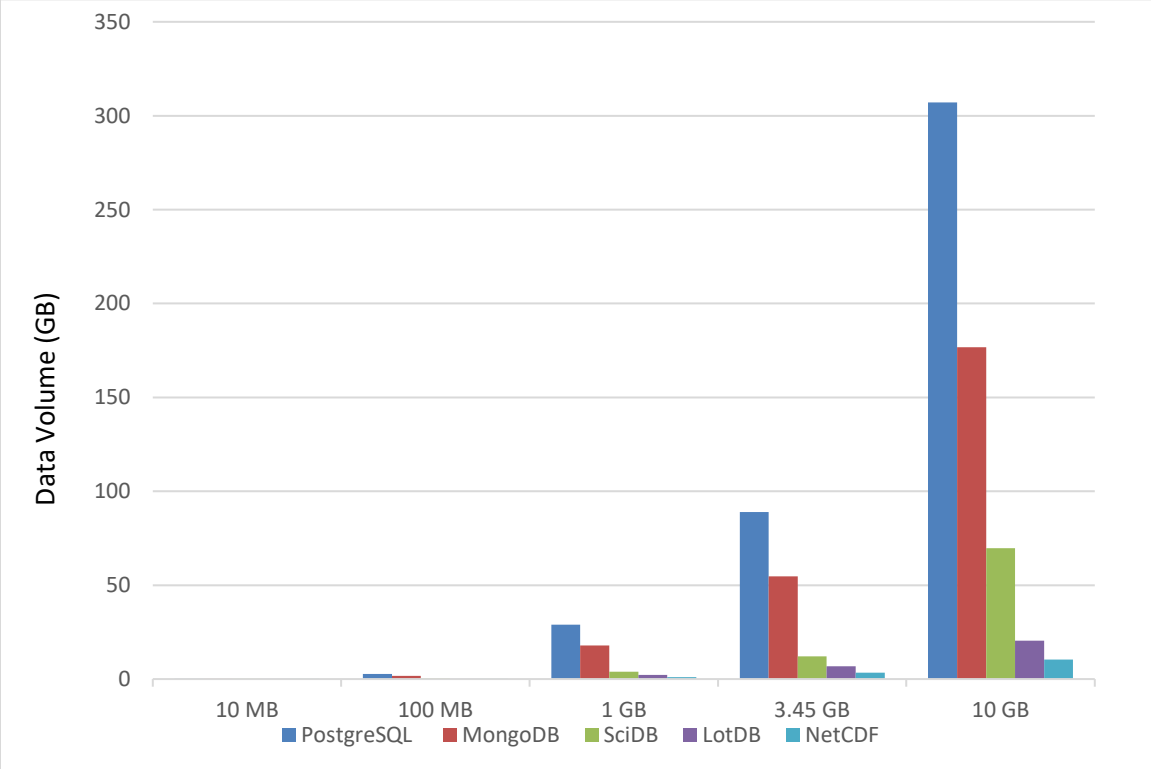


Figure 7 Data volume in different containers

The corresponding data storage volume in different containers for different cases is also recorded and displayed in Figure 7. The graph compares the difference in data expansion from raw data format to data storage volume in different databases. It is observed that although MongoDB took the longest time to upload data, it didn't use the most storage space; instead, PostgreSQL consumed the largest storage space in all cases, and data volume increased about 20 to 30 times from the raw data size. SciDB used much less space and has a data expansion rate of around 5. LotDB performed the best in all cases and has a 2 times data expansion rate; this meets the design of the unified storage structure and shows significant advantages in comparison to other tested databases. As data are stored in a unified storage structure in LotDB, no extra indexes or data chunks are needed. PostgreSQL and MongoDB are using a non-array data model to store arrays, extra storage for indexes is expected. SciDB uses an over-lapped chunking design, which costs additional storage space for the redundant part.

Table 4 Data Preprocessing & Uploading Time and Data Size in Containers

Container	Preprocessing Time (hours)	Uploading Time (hours)	Data Size (GB)
Raw Data (NetCDF format)	N/A	N/A	3.45
PostgreSQL 9.3 (SQL)	4	1.97	89
MongoDB 4.0.4 (NoSQL)	4	7.3	55
SciDB 18.1 (Array Database)	4	3.3	12.06
LotDB	0	0.03	6.78

Considering the performance for different databases on spatiotemporal queries, the MERRA-2 data set for the year 2017 was chosen. Table 4 lists the detailed data pre-processing and uploading time for this specific case. Among these data containers, MongoDB took the longest, and PostgreSQL required the largest space for data storage. By implementing the array as the fundamental data structure, SciDB and LotDB acquired much less storage. LotDB holds the top place both in data uploading time, and data storage size compare with the other three.

Figure 8 Spatiotemporal query run-time of PostgreSQL, MongoDB, SciDB and LotDB

Figure 8 illustrates the spatiotemporal query run-time across four containers for nine queries. In terms of performance, LotDB used the shortest time in all the queries

even compared with SciDB. The queries were designed to increase in complexity both spatially and temporally; the general run-time patterns of PostgreSQL and MongoDB are similar. They both tend to be stable in a certain range, the cost to do a simple query as Q1 is not much different from a complex query as Q9, although the number of data points retrieved varies largely. It implies that there exist some initial costs for each spatiotemporal query when executed in both databases. MongoDB is a document-based database, and PostgreSQL is a relational database. Both of them are not expected to have better performance than SciDB because their data models are mismatching with the array data model, which also agrees with the results from our previous studies on different data containers' abilities for handling big multidimensional array datasets (Hu et al. 2018). As a MongoDB insight, it tends to memory-map the whole collection into the physical memory of the machine during the query. When the total size of the dataset is significantly larger than the physical memory, a large number of memory page faults are observed and thus delayed query speed. As an array database, SciDB has much better performance outputs than PostgreSQL and MongoDB and proved to be one of the best databases on the market when handling multidimensional arrays. For LotDB, it has a significant time advantage among all selected databases. The reason behind the outstanding performance is because 1) LotDB implemented a native array data structure on the storage level, 2) memory-mapping technology is utilized for data retrieval on the byte level, and 3) query process is accelerated by the N-Dimensional hash function.

2.5.3 Indication of Results

The proposed database engine (LotDB) and it's associate algorithms work as designed to 1) providing fast query ability on gridded climate data, 2) reducing storage volume compared to other databases, and 3) presenting advancement of the indexing system and the integrated memory-mapping technology. The results have also proved the concept of developing a high-performance and cost-effective solution for gridded climate data management.

CHAPTER 3. A DICTIONARY BASED PRECIPITATION DOWNSCALING METHOD

3.1 Introduction

The studies of local climate impact are critical for environmental management (Sheehan et al. 2017), including applications such as water resources, ecosystem services, and agricultural productivity (Fatichi, Ivanov, and Caporali 2013; Yang et al. 2019). GCMs are our primary tools for simulating future climatological variables such as temperature, wind, and precipitation (Benestad 2016). However, GCM outputs are often too coarse for small-scale further analysis. A necessary step in local climate studies is the "downscaling" of climate variables that were generated at a lower resolution. Downscaling the low-resolution climate variables is used as a primary method to obtain high-resolution information (Wilby et al. 1998). For example, Regional Climate Models (RCMs) are nested in GCMs to produce higher resolution outputs in a dynamic downscaling fashion with limitations of computing-intensive and small study areas. Statistical downscaling methods are also developed to enhance GCM outputs' resolution directly to generate high-resolution results at low cost and with high efficiency.

Precipitation is one of the most challenging data in downscaling studies because precipitation events are not simulated well by models, and precipitation is not spatiotemporally continuous (Kundzewicz et al 2007; Maraun 2010; Vrac and Naveau 2007; Vandal 2018). Previous studies of downscaling precipitation can only provide

downscaled results as time-series predictions for a single station point or multi-points. Thus, traditional downscaling methods are insufficient for understanding the lifecycle of precipitation events in space, especially for short-duration events (Lenderink and Van Meijgaard 2008) than point scale or areal averages. However, downscaling precipitation as spatially distributed fields is a challenging task (Maraun 2010). It usually cannot be achieved by using statistical downscaling methods when high-resolution observation data are not available for the study area. The benefit of the proposed method is that it can downscale the precipitation of short duration events with full coverage in any region.

Statistical downscaling and dynamic downscaling are the two main methods to generate high-resolution regional precipitation data based on global climate model output or reanalysis data (Tang et al. 2016). Dynamical downscaling (Giorgi 1990; Wang et al. 2004) relies on the availability of GCM and climate model outputs. Statistical downscaling adopts the relationship between local climate variables and large-scale variables (Wilby et al., 1998) depending significantly on the accessibility to other climate variables and the target parameter such as precipitation. Stochastic downscaling can be realized without the consideration of other parameters, however, it yields high-resolution output only through mathematical transformations and ignores the natural features of the target. Machine learning methods are also proposed with the advancement of AI technology (Liu et al., 2019) but lack in considering the natural characteristics of precipitation.

I plan to fill this gap by proposing a novel downscaling method, called PreciPatch, to produce hourly precipitation at a higher spatial resolution based on GCM

outputs. The proposed method uses dictionary learning to construct a dictionary for relationships between low-resolution and high-resolution datasets based on historical data and applies the dictionary to downscaling low-resolution data for a certain timestamp. In addition, PreciPatch could be integrated with the research in Chapter 2 (LotDB) to accerlerate the downscaling process by replacing the lower level storage system with LotDB.

The remaining sections report our research as follows: Section 3.2 reviews related works in statistical downscaling, stochastic downscaling, Single Image Super-Resolution (SISR) based spatial downscaling, and types of downscaled precipitation products. Section 3.3 introduces the proposed methods in detail. Section 3.4 evaluates the proposed methods using several study areas across CONUS and compares results with results generated by other state-of-the-art downscaling methods.

3.2 Literature Review

Historically, downscaling is connected with the early development of numerical weather forecasting in the 1950s, when the commercially produced digital computers were stated to be widely used for business data processing. There are publications on the use of an empirical-statistical downscaling (ESD) technique, also known as the “analog method” or “analogue method”, in weather prediction since 1951 (Benestad 2016). The analogue method (AM) is a statistical method based on finding an analogue to a target climate variable from an earlier observation in a similar weather situation (Wetterhall, Halldin, and Xu 2005). The early climate models were only able to compute climate factors like atmospheric motion on a scale of hundreds of kilometers and with a limited

number of vertical levels. The term downscaling was possibly first used in a study by von Storch, Zorita, and Cubasch (1993). Statistical downscaling is always the main direction in precipitation downscaling; however, recent developments in Single Image Super-Resolution (SISR) have brought new opportunities into the downscaling domain. For example, SISR based downscaling methods do not need extra data when producing future predictions, which is a great advantage compares to statistical downscaling methods.

3.2.1 Statistical Downscaling of Precipitation

Downscaling could be addressed as a disaggregation problem, in which theoretically, any number of small-scale weather sequences can be associated with a given set of large-scale values (Wilks 2010). Statistical downscaling is popular in precipitation downscaling but the recent opportunities in Single Image Super-Resolution (SISR) for downscaling brought the advantage without requiring additional downscaling parameters such as temperature, pressure, wind speed and direction. Many studies have reviewed downscaling concepts and methods. For examples, Hewitson and Crane (1996) reviewed process-based and empirical methods. Wilby and Wigley (1997) investigated regression methods, weather pattern (circulation)-based approaches, stochastic weather generators, and limited-area climate models. Xu (1999) and Wilby et al. (2004) illustrated the guideline for statistical downscaling methods for climate scenarios. Fowler et al. (2007) discussed the hydrological community impact from downscaling, and Maraun et al. (2010) tried to connect dynamic downscaling with end-users. We review downscaling according to the methodology in the following subsections. There are some differences in classifying statistical downscaling methods; yet, general approaches could be divided into

the following groups: 1) regression models or Perfect Prognosis (PP) method, 2) weather pattern-based approaches, and 3) weather generators. Some popular methods are discussed in this section that covers those three categories but not limited to them. Although statistical downscaling methods vary one from another, the basic assumption is the same for all statistical downscaling methods: regional climates are largely a function of the large-scale atmospheric state (Fowler et al. 2007).

3.2.1.1 Perfect Prognosis (PP) Method

Perfect Prognosis (PP) method (Klein, Lewis, and Enger 1959) was first used to objectively forecast maximum and minimum temperature (Baker 1982). It relies on both large-scale observational data (often reanalysis datasets) and local-scale observations (Vandal 2018). The idea is to generate a statistical link between large-scale and local-scale observations and then transfer such a relationship to GCM simulation for future estimations (Maraun et al. 2010; Wong et al. 2014). Regression and analog methods are categorized as PP approaches. Some early researches have been done for climate change (Kim et al. 1984; Wigley et al. 1990) of local-scale monthly temperature and precipitation. Such approaches involve establishing linear or non-linear relationships between stationary parameters and coarse resolution gridded predictors (Wilby and Wigley 1997). The terrain difference (coastal and mountain influences) was found to have a significant impact on spatial and temporal variations in model performance. Landman et al. (2001) used the PP approach (Wilks 1995) for downscaling dynamical model quantities where the authors added an additional layer for bias correction. Their focus was seasonal forecasting of categorized streamflow for dams. Another example was

provided in 2004 by Tatli et al. to downscale monthly total precipitation over Turkey with 31 stations. The analog method was used to downscale daily precipitation from 2° resolution GCM output to point station data in Zambia (Brigadier et al. 2016). Chen, Brissette, and Leconte (2014) assessed linear regression-based methods in downscaling daily precipitation from GCM scale to RCM scale (45km and 15km) and also to a station scale across North America. Their results showed that the downscaling of daily precipitation occurrence was rarely successful at all scales, and they concluded that regression-based approaches did not have good performance in downscaling precipitation in the North America area (Chen, Brissette, and Leconte 2014). Part of the reason is that the potential deviations from real precipitation make it unsuitable in the PP method because it does not fit the crucial assumption of the PP approach (Maraun et al. 2010). Recent researches use machine learning to optimize the process of Optimal Predictor Selection (Najafi, Moradkhani, and Wherry 2010) and principal component analysis (Benestad et al. 2015).

3.2.1.2 Model Output Statistics (MOS) Method

Model Output Statistics (MOS) method was proposed and implemented in 1973 (Klein 1974; Bermowitz 1975) as a replacement of Perfect Prognosis (Benestad 2016). This technique uses the local-scale observations with the GCM outputs and does not necessarily rely on lower resolution observations (Vandal 2018). Turco et al. in 2011 tested the MOS implementation of the analog methodology (MOS analog) to downscale daily precipitation outputs over Spain. The method aimed to improve the RCM results instead of directly using GCM results. A combination of stochastic method and MOS was

proposed as a bias correction framework for daily precipitation generated by RCMs (Wong et al. 2014). Wetterhall et al. (2012) reviewed three MOS methods to adjust RCM daily precipitation, which are 1) a simple direct method (DM), 2) quantile-quantile mapping, and 3) a distribution-based scaling approach. The preprocessing step that used in that research involves direct interpolation, which potentially increased the bias before the downscaling process. Turco et al. (2017) introduced an analog-based MOS downscaling method for RCM bias correction, which preserves the original RCM climate change signal for different future periods. One limitation of MOS is that the MOS corrections are specific to the numerical models and cannot be used with other models (Maraun et al. 2010). Recent researches have shown a combination of MOS with support vector machine (SVM) to project spatial and temporal changes in rainfall (Pour et al. 2018).

3.2.1.3 Weather Pattern Approach

The weather pattern approach or automated weather classification was also referred to as the weather typing approach. This approach could be accomplished by deriving conditional probability distributions for observed data (e.g. station data). For example, the probability of a wet day following a wet day, or the mean wet-day amount associated with a given atmospheric circulation pattern (Hughes and Guttorp 1994). Conway and Jones (1998) reviewed three weather typing approaches based on Lamb weather types applied to the British Isles, and the methods could reproduce the monthly means of station rainfall time series during a validation period. Boé et al. (2006) implemented a multivariate downscaling method to generate local precipitation at

different sites in France. The distances of a given day to the different weather types were used as predictors. Satisfying results are achieved regarding low-frequency variation at daily timescale. Vrac, Stein, and Hayhoe (2007) proposed a nonhomogeneous stochastic weather typing approach to downscale precipitation at 37 rain gauges in Illinois state. Regional climate conditions were categorized into different types and the method modeled the transition probabilities from one weather pattern to another by a nonhomogeneous Markov model. Later researches have shown combinations of weather type studies with Extreme Value Theory (EVT) and Markov model (Vrac and Naveau 2007; Vrac, Stein, and Hayhoe 2007).

3.2.1.4 Weather Generators

Weather generators are statistical models used to create a long synthetic series of data, fill in the missing data and produce predictions (Hashmi, Shamseldin, and Melville 2009). The general idea is to mimic the effects of local processes (Field et al. 2014), assuming that these have no connection to global and large scales. The basic weather generators assume probability density function (pdf) is constant, with random fluctuations (Benestad 2016). For example, the Richardson's (1981) WGEN model was designed to simulate daily time-series of precipitation amount, maximum and minimum temperature, and solar radiation for the present climate (Wilby and Wigley 1997). Wilby et al. (2002) explored the use of synoptic-scale predictors to downscale the components of daily precipitation at sites across the British Isles. The conditional models used in the study displayed a greater skill for monthly rainfall statistics to a controlled model; however, the generality of the results should be further examined. Chen, Brissette, and

Leconte (2012) presented a statistical downscaling approach for seasonal and annual precipitation changes using stochastic weather generator and the change factor on a Canadian watershed, which showed advantages in simulating low flows. By coupling a single-site downscaling method and a multi-site weather generator, Chen, Chen, and Guo (2018) proposed a downscaling method for multi-station sites' precipitation time series, which is an efficient method and can be used to investigate the spatial variability of climate change impacts regarding precipitation. Major studies in weather generator downscaling area are focusing on the site-based generation of long time-series predictions. Recent studies have shown an increase in temporal resolution and extending the single-site approach to multi-sites (Peleg et al. 2019; Verdin et al. 2018; Sørup et al. 2016).

3.2.1.5 Limited-Area Climate Models (LAMs)

Limited-area climate models (LAMs) were proposed to embed a higher-resolution limited-area climate model (LAM) within GCM in early times, and it has been referred to as the RCM in the later studies. However, one concern is that LAM rainfall estimates are not reliable on small scales due to the incomplete numerical representation of small-scale processes and limited resolution of the observation networks (Rebora et al. 2006). Recent researches do not consider this method as one of the statistical downscaling methods anymore (Benestad 2016; Vandal 2018; Noor et al. 2019).

3.2.1.6 Integrated Approaches

Although there are many downscaling methods applied to GCM outputs, in reality, downscaling approaches often embrace more than one technique and therefore

tend to be a hybrid in nature (Wilby and Wigley 1997). For example, RCM is used to dynamically downscale outputs from a few GCMs, and statistical relationships between the RCM outputs and GCM patterns are usually used to downscale other GCMs (Walton et al. 2015). Major researches in the statistical downscaling field are often focusing on specific areas and leads to case-by-case studies, and therefore, the downscaled quality is highly dependent on the availability of local observation data.

3.2.1.7 Limitations of Statistical Downscaling

Most statistical downscaling methods rely on empirical mathematical relationships from coarse resolution predictors to finer resolution predictands. These relationships are often much faster to implement and compute than dynamical downscaling, which is the main reason for using statistical downscaling. However, such relationships are subject to the stationarity assumption that the relationship between predictors and predictands continues to hold, even in a changing climate (Walton et al. 2015; Wilby and Wigley 1997). Practically, statistical downscaling is coupled with dynamic downscaling methods to provide more reliable results (Chen, Brissette, and Leconte 2012). Both dynamic and statistical downscaling methods need assumptions that cannot be verified in a climate change context (Giorgi et al. 2001). However, several criteria can be used to assist in the selection of the most suitable approach depending on the application (Wilby et al. 2004). One major critique of the consistency between GCM inputs and downscaled outputs is that the downscaled output is constrained by the limitations of the GCM input. A single GCM may give a misleading picture of the true state of knowledge about climate change, including in the region of interest. The downscaling results of this single GCM will

likewise be misleading (Walton et al. 2015). Meanwhile, with increasing demands from local climate researches for better quality downscaled results and higher requirements for downscaled outputs, statistical downscaling can hardly meet the needs of providing a data independent downscaling method to simulate precipitation as precipitation fields at a local scale with a high temporal resolution. Another limitation of statistical downscaling, for example, is that it can only provide downscaling results for which the input GCM product is corresponding to observation (Benestad 2016). Climate variables like precipitation are not well simulated in either GCM or RCM, therefore it is difficult to downscale through statistical methods. Meanwhile, most statistical methods are area limited and require the availability of higher resolution observation data (Jarosch, Anslow, and Clarke 2012), while it is challenging for many regions to get observation data (e.g. mountain areas). Additionally, statistical methods can hardly produce precipitation field output, but the precipitation field is essential for driving distributed hydrological models for climate change impact studies (Chen, Chen, and Guo 2018). Most researches in statistical downscaling often focus on specific areas and lead to case-by-case studies. The downscaled quality is highly dependent on the availability of local observation data. Specifically, traditional statistical downscaling methods have the following limitations: 1) their dependency on higher-resolution observations or other variables in addition to precipitation, 2) their results vary in quality across different regions and often not in the form of precipitation fields, and 3) their ability to downscale future estimations of precipitation is limited to an hourly basis.

3.2.2 Stochastic Downscaling

Stochastic downscaling is sometimes classified into the weather pattern approach because of its similar assumption of probability distributions. A more recent recognition is to treat the stochastic downscaling methods as disaggregation techniques that could generate small-scale ensemble rainfall predictions (Rebora et al. 2006; Posadas et al. 2015; Terzago, Palazzi, and Hardenberg 2018). In operations, the stochastic downscaling method is one of the few techniques available that could produce downscaled precipitation output as precipitation fields. Rainfall Filtered Autoregressive Model (RainFARM) is one of the most successful stochastic downscaling models (Rebora et al. 2006), which is based on the nonlinear transformation of a Gaussian random field and constrained by rainfall amounts at larger scales. Many have attempted to enhance the RainFARM. For example, Terzago, Palazzi, and Hardenberg (2018) added a map of extra weights to adjust RainFARM output. Another approach is done by Posadas et al. (2015) using a customized disaggregation model, which requires extra station data to correct the final results in the form of generating local heterogeneity. One major argument for using stochastic downscaling is its random process, in which random generated downscaled results cannot be taken as a faithful deterministic prediction of small-scale precipitation and could only be treated as a realization of a process with the appropriate statistical properties (Rebora et al. 2006). At the same time, stochastic downscaling methods fail to simulate precipitation events on the regional level at high temporal resolution (e.g. hourly). The long-term accumulated weights used in stochastic downscaling methods do not apply to hourly events and are not yet tested against short-term precipitation events.

3.2.3 Single Image Super-Resolution (SISR) Based Spatial Downscaling

Traditional statistical downscaling methods rely heavily on datasets in addition to GCMs, mostly stationary observations. A range of statistical downscaling models, from early regressions to neural networks, have been developed for regions where model calibration is possible because of the availability of sufficiently good datasets (IPCC n.d.). Machine learning frameworks and algorithms are not completely new in the downscaling area. Some integrations of statistical downscaling and machine learning methods include optimizing the process of Optimal Predictor Selection (Najafi, Moradkhani, and Wherry 2010), principal component analysis (Benestad et al. 2015), and using Random Forests (RF) in PP method (He 2016). However, there are very few researches conducted outside the statistical downscaling category.

Recent developments in the image processing field have shown promising performance in increasing spatial resolution of a single image through prior knowledge learned using machine learning approaches (Dong et al. 2014; Kim et al. 2016; Zhang et al. 2018). By adopting similar techniques or ideas and using them in downscaling applications, Super-Resolution (SR) algorithms and frameworks have drawn significant attention because of their potentials. Dictionary learning-based method, or sparse regularization (Ebtehaj et al. 2012) and Super-Resolution Convolutional Neural Networks (SRCNN) based method (Vandal et al. 2017) are two main directions in precipitation downscaling applications. Dictionary learning is an old method while SRCNN is newer and is one of the most popular methods for super-resolution reconstruction in image processing. Although applying SISR algorithms to gridded datasets and downscaling

precipitation fields have been attempted by a few researchers (Vandal et al. 2017), providing an appealing result is still a difficult task. Recent developments in machine learning and deep learning have shown promising results in image reconstruction with super-resolution algorithms, but not enough to produce a satisfying result for precipitation downscaling purposes. One drawback of this method is it involves the process of image interpolation, which is not a problem for images but dramatically increased the uncertainties of gridded datasets. Also, SRCNN based method such as DeepSD has added terrain information to help generate precipitation distribution patterns on 2D space; however, it is not often applicable for short-duration precipitation events. At the same time, "end users" of the downscaled products usually have higher requirements than just sharpening the edges or increasing spatial resolution from 2 to 4 times (at least 8 to 12 times is more common in the downscaling field), which is beyond the capability of current SISR algorithms.

3.2.4 Types of Outputs from Precipitation Downscaling

Precipitation downscaling could provide three types of outputs: 1) point scale, 2) areal average and 3) spatially distributed precipitation fields (Maraun 2010). Specifically, point scale products refer to downscaled outputs as the prediction of station sites' observation or fitting GCM output to station networks. Areal average products are usually in forms of maps of precipitation zones with high-density station coverage, for example, gridded station maps (Hewitson and Crane 2006), analog maps (Wang and Zhang 2008), and climate zone maps (Chandler 2011). The temporal resolution for this is often monthly, seasonally, or annually. Spatially distributed precipitation field refers to

the continuous rainfall field products and is the most challenging one to produce. It simulates fields of precipitation that can provide continuous spatial information as input to distributed hydrological models (Terzago, Palazzi, and Hardenberg 2018). The temporal resolution of this type is less than a day (e.g. hourly), thus able to capture short-term precipitation events. Traditional statistical downscaling methods usually provide point scale results (e.g. time series prediction of one weather station or multi stations in an area), or areal average. Very few of the researches were aimed to provide spatially distributed precipitation fields, i.e. regular girds products (Rebora et al. 2006; Posadas et al. 2015; He 2016; Vandal et al. 2017).

3.3 Methodology

The development of the PreciPatch is inspired by the idea of the Analog Method (AM) and sparse regularization or, more recently called, dictionary learning. The general assumption is made that future weather will behave similarly to the past. By constructing a dictionary for low-resolution and high-resolution patches and using the dictionary with mapping information from historical data, the downscaling process could be achieved through a dictionary learning approach.

3.3.1 Precipitation Downscaling Based on Dictionary Learning

3.3.1.1 Dictionary Learning for Super-Resolution

A classical dictionary trained for sparse coding consists of a limited collection of static atoms, i.e. a set of sample-based fixed patches. It is not a problem in the image super-resolution field since the goal for image enhancement is different from precipitation downscaling (sharpen the image vs. create local variation), and the

resolution enhancement rates are different (2 to 4 vs. 8 to 12) (Vandal 2018). In a standard dictionary learning approach for image super-resolution, the estimation of a high-resolution image, $x = [x_1, x_2, x_3, \dots, x_m]^T \in \mathbb{R}^m$, from its low-resolution counterpart, $y \in \mathbb{R}^n$, where $n \leq m$, can be recast as an inverse problem. x may be estimated from y through a linear structured degradation operator, $H \in \mathbb{R}^{n \times m}$ and a noise e (Ebtehaj et al. 2012):

Equation 2 Inverse estimation

$$y = Hx + e$$

The sparsity of x implies that it can be approximated by its orthogonal projection x_S by a few atoms, $\{\emptyset_i\}_{i=1}^{M}$ of a dictionary, $\Phi \in \mathbb{R}^{m \times M}$ with a vector of representation coefficient c. By approximating x with x_S, the equation could be rewritten as:

Equation 3 Inverse estimation with a dictionary

$$y = H\Phi c + e', where\ e' = H(x - x_S) + e$$

y could also be approximated by using a low-resolution dictionary. The idea of dictionary learning is to train two dictionaries: one low-resolution dictionary and one high-resolution dictionary. By solving the approximation problem for low-resolution, the high-resolution can be recovered using the gathered information. The explanation above is a highly simplified version of sparse coding and dictionary learning, but it could be accomplished in many ways. For example, the SRCNN super-resolution method could be treated as a revised version of dictionary learning with modifications under a deep learning framework (Dong et al. 2014).

3.3.1.2 Dynamic Dictionary Learning

A critical aspect of precipitation downscaling is to generate local variations from large-scale areal averages. Ideally, the atom number in a dictionary should be large enough to create variations under complex conditions. The generality of large-scale and local variables could be achieved by merging closer patches, regularizing patches, or using regularized searching functions (Freeman, Jones, and Pasztor 2002). The dictionary learning method is still inadequate in precipitation downscaling circumstances. One important reason is that gridded data sets are different from pure image files, and variations in precipitation are difficult to simulate and, therefore hard to generalize.

The proposed approach is inspired by the idea of dictionary learning; however, it differs in many ways and does not follow any dictionary learning frameworks. For example, the atoms in the dictionary are not used directly as patches; instead, they were used as base patch values to simulate new patches. Therefore, the dictionary learning process becomes a dynamic approach.

3.3.1.3 Downscaling Workflow

The general workflow for the proposed downscaling method is to construct a dictionary consisting of linked low and high-resolution patches using historical datasets. A similarity search is then performed on the low-resolution patch with the low-resolution GCM product as input to find the linked high-resolution patch in the dictionary. The high-resolution output will be produced by linearly aligning the corresponding high-resolution patches. In detail, the dynamic dictionary learning approach includes the following steps:

1. Select a GCM precipitation product that needs to be downscaled.

2. Select a downscaling scale ratio or a desired spatial resolution. Let L_s denote the scaling factor, for $L_s \in \mathbb{Z}^+$.
3. Select an existing gridded precipitation dataset A_{HR} (observation and bias-corrected data are preferred) that has a native spatial resolution match or close to the desired spatial resolution L_s. The temporal resolution should also be matched on the same level (e.g. hourly, 6-hourly, daily, etc).
4. Upscale A_{HR} to the spatial resolution level that matches or close to the GCM precipitation product (a widely used method is to aggregate grids as averages from the higher resolution (He 2016)).
5. Use the upscaled output and A_{HR} as inputs to construct the patch dictionary.
6. Use the GCM precipitation product as input and perform a similarity search on the constructed dictionary to find an estimated high-resolution patch for each grid.
7. Align each high-resolution patch to form a weight mask for the high-resolution precipitation field.
8. Use the weight mask as a basis, and the original GCM product as the rain rate constraint to produce the downscaled output.

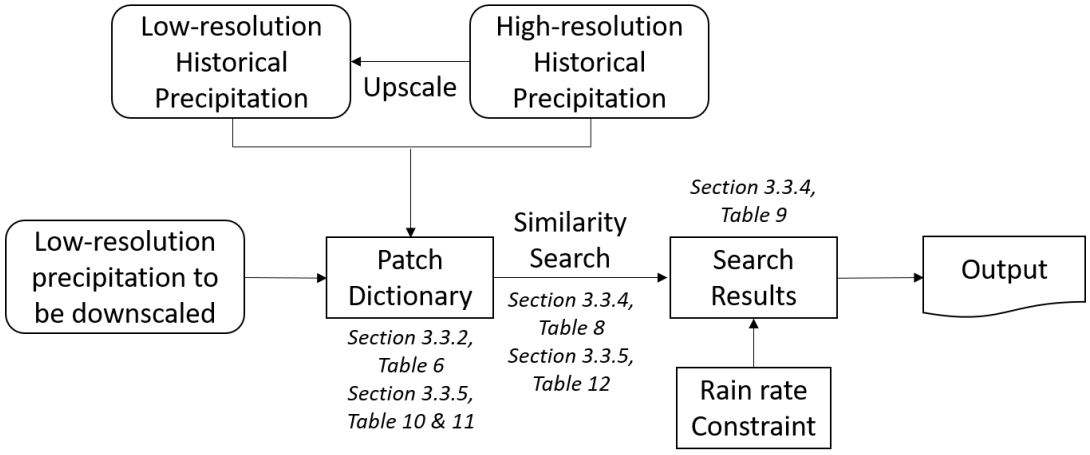


Figure 9 General workflow of PreciPatch

The workflow of the PreciPatch method is shown in Figure 9 and detailed in the following subsections.

3.3.2 High-Resolution Dictionary Construction

The core of the proposed dynamic dictionary learning approach is to construct a dictionary consisting of both low-resolution and high-resolution patches. Patches are defined as subsets from the original gridded datasets, and the dictionary is enriched by ingesting more datasets. This section introduces algorithms to construct the dictionary.

3.3.2.1 Recording the Spatial Difference

Before constructing the dictionary, the Diff Array is proposed and defined in Table 5 as a record of space difference between grid cells, and is used in further analysis of other algorithms. Figure 10 shows the construction of the Diff Array; the difference between cells is defined as “top minus bottom” and “left minus right”. In the Diff Array, precipitation value differences between nearby grids are recorded and will be used to classify the spatial differences between grids. The Diff Array stores 12 values in total for

a three by three array. The Diff Array will be used for dictionary construction (section 3.3.2.2) and similarity search (section 3.3.4).

Table 5 The Diff Array

Name: Create the Diff Array
Input: Array $B_{LR}[3][3]$, a non-empty array with size 3×3 **Output**: $\text{Diff}(B_{LR}[3][3]) = B_{DIFF}[12]$, the Diff array of $B_{LR}[3][3]$ with size 12 **Procedure**: (1) $\text{Diff}(B_{LR}[3][3]) = B_{DIFF}[12]$ is defined as: $B_{DIFF}[0] = B_{LR}[0][0] - B_{LR}[0][1]$, $B_{DIFF}[1] = B_{LR}[0][1] - B_{LR}[0][2]$, $B_{DIFF}[2] = B_{LR}[0][0] - B_{LR}[1][0]$, $B_{DIFF}[3] = B_{LR}[0][1] - B_{LR}[1][1]$, $B_{DIFF}[4] = B_{LR}[0][2] - B_{LR}[1][2]$, $B_{DIFF}[5] = B_{LR}[1][0] - B_{LR}[1][1]$, $B_{DIFF}[6] = B_{LR}[1][1] - B_{LR}[1][2]$, $B_{DIFF}[7] = B_{LR}[1][0] - B_{LR}[2][0]$, $B_{DIFF}[8] = B_{LR}[1][1] - B_{LR}[2][1]$, $B_{DIFF}[9] = B_{LR}[1][2] - B_{LR}[2][2]$, $B_{DIFF}[10] = B_{L}[2][0] - B_{LR}[2][1]$, $B_{DIFF}[11] = B_{LR}[2][1] - B_{LR}[2][2]$ (2) **Return** $B_{DIFF}[12]$

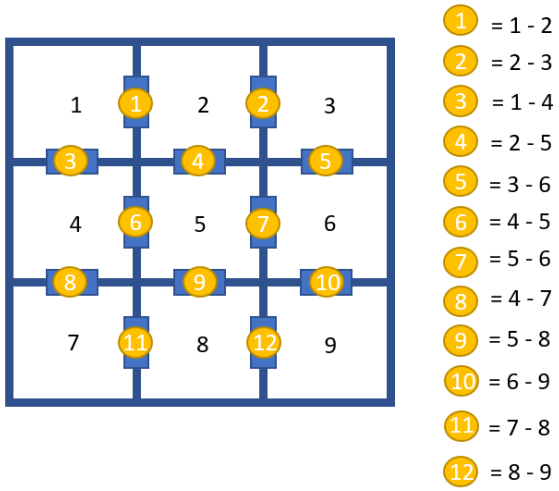


Figure 10 A graph representation of Diff Array

3.3.2.2 The Dictionary Construction Algorithm

The patch dictionary used in this downscaling method is constructed using the algorithm introduced in Table 6. The high-resolution and low-resolution dictionaries are integrated as one, and the spatial information (i.e., Diff Array) of the low-resolution patches is stored. High-resolution datasets and low-resolution datasets are decomposed into small patches and reorganized into a patch dictionary. In this case, the low-resolution datasets are divided into three by three low-resolution patches, and the high-resolution datasets are divided into high-resolution patches with sizes corresponding to the scaling factor. The Diff Array of each low-resolution patch is also calculated and stored, which provides additional information about the spatial relationships between nearby grids of each low-resolution patch.

Table 6 The dictionary construction algorithm

Name: Construction of the high-resolution dictionary

Definition:

L_s is the scaling factor, for $L_s \in \mathbb{Z}^+$

A_{HR} is one time slice from a gridded precipitation dataset

A_{LR} is the aggregated (upscaled) result from A_{HR} using area average based on the scaling factor L_s

Input:

Array $A_{LR}[m][n] = \begin{bmatrix} l_{0,0} & \cdots & l_{0,n} \\ \vdots & \ddots & \vdots \\ l_{m,0} & \cdots & l_{m,n} \end{bmatrix}$, for $m, n \in \mathbb{Z}^+$

Array $A_{HR}[q][p] = \begin{bmatrix} h_{0,0} & \cdots & h_{0,p} \\ \vdots & \ddots & \vdots \\ h_{q,0} & \cdots & h_{q,p} \end{bmatrix}$, for $q, p \in \mathbb{Z}^+, q = (m+1) * L_s - 1, p = (n+1) * L_s - 1$

Output:

$D\{d_{LR}, d_{HR}, d_{DIFF}\}$, the patch dictionary consists of the low-resolution patch set d_{LR}, high-resolution patch set d_{HR}, and spatial difference set d_{DIFF} (defined in table 5)

Procedure:

(1) **For** ($j = 1$ **to** $j = m - 1$) **do** {
  For ($i = 1$ **to** $i = n - 1$) **do** {
    New Array $B_{LR}[3][3]$, Array $B_{HR}[L_s][L_s]$, Array $B_{DIFF}[12]$
    For ($k = -1$ **to** $k = 1$) **do** {
      For ($r = -1$ **to** $r = 1$) **do** {
        $B_{LR}[k+1][r+1] = A_{LR}[j+k][i+r]$
      }
    }
    Appending B_{LR} as the last element of d_{LR}
    $B_{DIFF}[12] = \text{Diff}(B_{LR}[3][3])$
    Appending B_{DIFF} as the last element of d_{DIFF}
    $B_{HR} = A_{HR}[(j * L_s)$ **to** $((j+1) * L_s - 1)][(i * L_s)$ **to** $((i+1) * L_s - 1)]$
    Appending B_{HR} as the last element of d_{HR}
  }
}

(2) **Return** $D\{d_{LR}, d_{HR}, d_{DIFF}\}$ as the integration of sets d_{LR}, d_{HR}, and d_{DIFF}

3.3.3 A Dynamic Time Warping (DTW) Based Similarity Search

After constructing the dictionary, a similarity search is performed against the dictionary using low-resolution inputs. It is different from the widely used Mean Squared Error (MSE) or Euclidean Distance-based similarity search. Dynamic Time Warping (DTW) distance is utilized in this research to help to measure spatial similarities between two low-resolution patches. For example, the following algorithm is used to calculate the DTW distance of two data series with equal length. The original version of DTW distance is used for measuring the similarity between two-time series with different lengths. See related research done by Itakura (1975) and Keogh and Ratanamahatana (2005). Operational speaking, the DTW distance could be applied to any data series in the form of arrays. Table 7 provides the algorithm for an application, revised from the original version of DTW (Keogh and Ratanamahatana 2005).

Table 7 DTW distance review

Name: DTW distance of two data series with equal length (*n*)
Input: Array $A = [a_1, a_2, \dots, a_i, \dots, a_n]$ Array $B = [b_1, b_2, \dots, b_j, \dots, b_n]$ **Output**: $distance_{DTW}(A, B)$, the DTW distance between A and B **Procedure**: (1) **Construct** an *n*-by-*n* matrix where the (i^{th}, j^{th}) element of the matrix contains the distance $dist(a_i, b_j)$ between the two points a_i and b_j, (i.e. $dist(a_i, b_j) = (a_i - b_j)^2$). (2) A warping path W is a contiguous set of matrix elements that defines a mapping between A and B. The k^{th} element of W is defined as $w_k = (i, j)_k$, therefore, $W = w_1, w_2, \dots, w_k, \dots, w_K$, where $n \leq K \leq 2n - 1$ W is subject to several constraints: • Boundary conditions: $w_1 = (1, 1)$ and $w_K = (n, n)$

- Continuity: given $w_k = (c, d)$, then $w_{k-1} = (c', d')$, where $c - c' \leq 1$ and $d - d' \leq 1$
- Monotonicity: given $w_k = (c, d)$, then $w_{k-1} = (c', d')$, where $c - c' \geq 0$ and $d - d' \geq 0$

(3) The DTW path is the path that minimizes the warping cost:

$$DTW(A, B) = \min \{ \sqrt{\sum_{k=1}^{K} w_k} \}$$

(4) Define the cumulative distance $\gamma(i, j)$ as the distance found in the current cell and the minimum of the cumulative distances of the adjacent elements:

$$\gamma(i, j) = dist(a_i, b_j) + \min \{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}$$

(5) $distance_{DTW}(A, B) = \gamma(n, n)$, where the path is $DTW(A, B)$

(6) **Return** $distance_{DTW}(A, B)$

3.3.4 Generate the High-Resolution Weight Mask Through a Double-Layer DTW Similarity Fuzzy Search Algorithm

DTW distance offers a more comprehensive estimation for similarity than simple MSE. This approach could capture linear changes; however, precipitation field downscaling requires estimation of spatial changes. A double-layer DTW similarity search is then proposed in Table 8, which utilizes the Diff Array stored in the previous steps to compare rainfall peaks and valleys in the neighborhood cells spatially. Additionally, a fuzzy search layer is added into the algorithm to avoid the overfitting of patches. In Table 8, each grid from the low-resolution input is compared with the patches in the constructed dictionary along with its spatial relationship with nearby eight grids (the Diff Array). The final high-resolution patches will be estimated by comparing the similarity of the low-resolution input grids with low-resolution dictionary patches. Specifically, three closest patches from the dictionary are selected for each input grid,

and their corresponding high-resolution patches are averaged to produce a final high-resolution patch for the input grid. The final high-resolution output field is produced by aligning each high-resolution patch found for each input grid.

Table 8 A double-layer DTW similarity fuzzy search algorithm

Name: Similarity fuzzy search based on a double-layer DTW distance

Input:

L_s, for $L_s \in \mathbb{Z}^+$, is the scaling factor

$D\{d_{LR}, d_{HR}, d_{DIFF}\}$, is the patch dictionary with size T_D for each set

Array $C_{LR}[y][x] = \begin{bmatrix} l_{0,0} & \cdots & l_{0,x} \\ \vdots & \ddots & \vdots \\ l_{y,0} & \cdots & l_{y,x} \end{bmatrix}$, for $y, x \in \mathbb{Z}^+$ and $y, x \geq 2$, is the low-resolution dataset

Output:

Array $C_{HR}[g][k] = \begin{bmatrix} h_{0,0} & \cdots & h_{0,k} \\ \vdots & \ddots & \vdots \\ h_{g,0} & \cdots & h_{g,k} \end{bmatrix}$, for $g, k \in \mathbb{Z}^+, g = (y-1) * L_s - 1, k = (x-1) * L_s - 1$, is the downscaled weight mask

Procedure:

(1) **For** ($j = 1$ **to** $j = y - 1$) **do** {
  For ($i = 1$ **to** $i = x - 1$) **do** {
    New Array $E_{LR}[3][3]$, Array $E_{DIFF}[12]$,
    New Array $E_{1st}[L_s][L_s]$, Array $E_{2nd}[L_s][L_s]$, Array $E_{3rd}[L_s][L_s]$
    For ($k = -1$ **to** $k = 1$) **do** {
      For ($r = -1$ **to** $r = 1$) **do** {
        $E_{LR}[k+1][r+1] = C_{LR}[j+k][i+r]$
      }
    }
    $E_{DIFF}[12] = \text{Diff}(E_{LR}[3][3])$
    Let 1st_best_so_far = infinity
    Let 2nd_best_so_far = infinity
    Let 3rd_best_so_far = infinity
    For each t **in** T_D **do** {
      $dist_{part1} = distance_{DTW}(flat(E_{LR}), flat(d_{LR}[t]))$
      $dist_{part2} = distance_{DTW}(E_{DIFF}, d_{DIFF}[t])$
      $dist_{all} = dist_{part2} + dist_{part2}$
      If ($dist_{all} <$ 1st_best_so_far) **then** {
        1st_best_so_far $= dist_{all}$

$\mathrm{E}_{1st} = d_{HR}[t]$ }
Else if ($dist_{all} <$ 2nd_best_so_far) **then** {
2nd_best_so_far $= dist_{all}$
$\mathrm{E}_{2nd} = d_{HR}[t]$ }
Else if ($dist_{all} <$ 3rd_best_so_far) **then** {
3rd_best_so_far $= dist_{all}$
$\mathrm{E}_{3rd} = d_{HR}[t]$ }
Else {**Continue**}
}
$\mathrm{C}_{HR}[((j-1)*\mathrm{L}_s)$ **to** $(j*\mathrm{L}_s-1)][((i-1)*\mathrm{L}_s)$ **to** $(*\mathrm{L}_s-1)] = Average(\mathrm{E}_{1st}+\mathrm{E}_{2nd}+\mathrm{E}_{3rd})$
}
}
(2) **Return** C_{HR}

The result from the algorithm in Table 8 is a high-resolution weight mask, but it cannot be treated as the final downscaling result, yet it is very close to the aimed output. The downscaling algorithm described in Table 9 is a necessary step to make sure the final output is constrained by the original low-resolution input, i.e., consistent with the large-scale precipitation amount.

Table 9 Downscaling algorithm based on a weighted mask and constrains

Name: Produce downscaled results from a weighted mask and constrains
Input: L_s, for $\mathrm{L}_s \in \mathbb{Z}^+$, is the scaling factor Array $\mathrm{C}_{LR}[y][x] = \begin{bmatrix} l_{0,0} & \cdots & l_{0,x} \\ \vdots & \ddots & \vdots \\ l_{y,0} & \cdots & l_{y,x} \end{bmatrix}$, for $y, x \in \mathbb{Z}^+$ and $y, x \geq 2$, is the low-resolution dataset Array $\mathrm{C}_{HR}[g][k] = \begin{bmatrix} h_{0,0} & \cdots & h_{0,k} \\ \vdots & \ddots & \vdots \\ h_{g,0} & \cdots & h_{g,k} \end{bmatrix}$, for $g, k \in \mathbb{Z}^+, g = (y-1)*\mathrm{L}_s - 1, k = (x-1)*\mathrm{L}_s - 1$, is the downscaled weight mask

Output:

Array $DS_{HR}[g][k] = \begin{bmatrix} h_{0,0} & \cdots & h_{0,k} \\ \vdots & \ddots & \vdots \\ h_{g,0} & \cdots & h_{g,k} \end{bmatrix}$, for $g, k \in \mathbb{Z}^+, g = (y-1) * \mathrm{L}_s - 1, k = (x-1) * \mathrm{L}_s - 1$, is the downscaled precipitation field from C_{LR}

Procedure:

(1) **For** ($j = 1$ **to** $j = y - 1$) **do** {

For ($i = 1$ **to** $i = x - 1$) **do** {

$\mathrm{DS}_{HR}\left[\left((j-1) * \mathrm{L}_s\right)\ \textbf{to}\ (j * \mathrm{L}_s - 1)\right]\left[\left((i-1) * \mathrm{L}_s\right)\ \textbf{to}\ (i * \mathrm{L}_s - 1)\right] =$

$Sum(\mathrm{C}_{LR}[j][i]) * \left(\frac{\mathrm{C}_{HR}[((j-1)*\mathrm{L}_s)\ \textbf{to}\ (j*\mathrm{L}_s-1)][((i-1)*\mathrm{L}_s)\ \textbf{to}\ (i*\mathrm{L}_s-1)]}{Sum(\mathrm{C}_R[((j-1)*\mathrm{L}_s)\ \textbf{to}\ (j*\mathrm{L}_s-1)][((i-1)*\mathrm{L}_s)\ \textbf{to}\ (i*\mathrm{L}_s-1)])}\right)$

}

}

(2) **Return** DS_{HR}

3.3.5 Patch Dictionary Classification and Loose Index

The spatiotemporal complexity of DTW is $O(n^2)$ for data series with the same length n (Keogh and Ratanamahatana 2005). A patch dictionary contains K data series would have a complexity of $O(K \cdot n^2)$. The databases or dictionary used in prior research is relatively small, typically less than 10,000 samples (Tan, Webb, and Petitjean 2017). However, the dictionary constructed in previous sections will easily exceed 1,000,000 samples after absorbing years of data. Therefore, classification and indexing become necessary if we want to maintain searching performance at a practical level. The following sections will introduce one classification algorithm to divide the patch dictionary into sub-dictionaries, an effective loose index method to generate an individual index for each sub-dictionary, and an updated similarity search that utilizes the classification and indexing system.

3.3.5.1 Dictionary Classification Algorithm

The patch classification algorithm in Table 10 is a simple but effective coding method based on the spatial difference of low-resolution patches. Sub-dictionaries are classified according to the code. A similarity search will be done against individual sub-dictionary instead of the whole dictionary, which saves tremendous scan time.

Table 10 Patch dictionary classification algorithm

Name: Build up classification

Input:
$D\{d_{LR}, d_{HR}, d_{DIFF}\}$, is the patch dictionary with size T_D for each set
Output:
The classification code $Code_D$ of an element $D\{d_{LR}[t], d_{HR}[t], d_{DIFF}[t]\}$ in $D\{d_{LR}, d_{HR}, d_{DIFF}\}$ for an element t in T_D
Procedure:
(1) **Let** $Code_D =$ ""
(2) **For each** d **in** $d_{DIFF}[t]$ **do** {
If ($d < 0$) **then** $\{Code_D += string(9)\}$
Else if ($d > 0$) **then** $\{Code_D += string(1)\}$
Else if ($d = 0$) **then** $\{Code_D += string(0)\}$
Else {**print (**NaN**)**}
}
(3) **Return** $Code_D$

Figure 11 gives a preview of the classification system when implementing. Each file folder in the figure represents a sub-dictionary that has been classified according to the encoding system.

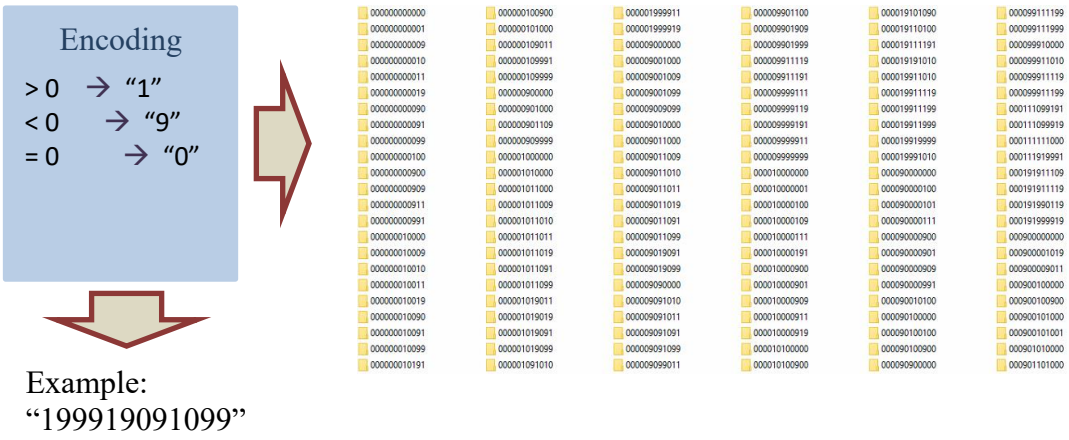


Figure 11 Dictionary classification

3.3.5.2 Loose Index Algorithm for Sub-Dictionaries

Although the atom number in sub-dictionaries is much less than the original dictionary, the scanning process will still be long if large amounts of historical data are used in constructing the dictionary. Therefore, an indexing system is always needed to reduce lookup time. An index is introduced in Table 11 to store comparative DTW distance between the patches and the x-axis (sets with all zeros). The idea is to filter potential solutions for arbitrary similarity searches. In Table 11, the combined DTW distance is pre-calculated for each low-resolution patch between the zero-vector and its Diff Array between the zero-vector. The combined DTW distance could not be treated as the exact index but enough for a loose indexing system.

Table 11 Sub-dictionary loose indexing algorithm

Name: Build up a loose index for the sub-dictionary
Input: $SD\{sd_{LR}, sd_{HR}, sd_{DIFF}\}$, is the sub-dictionary under a classification code $Code_{SD}$

with size T_{SD} for each set
Output:
Index set sd_{index} with size T_{SD}
Procedure:
(1) **For each** t **in** T_{SD} **do** {
$dist_{part1} = distance_{DTW}(flat(sd_{LR}[t]), flat(zeros(sd_{LR}[t])))$
$dist_{part2} = distance_{DTW}(sd_{DIFF}[t], zeros(sd_{DIFF}[t]))$
$dist_{zero} = dist_{part2} + dist_{part2}$
Appending $dist_{zero}$ as the last element of sd_{index}
}
(2) **Return** sd_{index}

3.3.5.3 Updated Similarity Search Based on Dictionary Classification and Loose Index

After classifying the dictionary and building the loose index, the similarity search algorithm should be revised to leverage the index structure. Table 12 offers the updated version part of the double-layer similarity fuzzy search algorithm from Table 8. In Table 12, each low-resolution input grid is compared with the patches in the corresponding sub-dictionaries that follows the same spatial relationship with its 8 nearby grids. The DTW distance between the patches and the zero-vector haven been pre-calculated and is compared with the DTW distance between the input grid with the zero-vector to form a candidate set of patches. Then the exact DTW distance is calculated among the candidate patches to find the three closest patches and produce the final patch piece.

Table 12 Similarity search algorithm based on dictionary classification and loose index

Name: using the classification and loose index to search
Input: L_s, for $L_s \in \mathbb{Z}^+$, is the scaling factor N_{patch}, for $N_{patch} \in \mathbb{Z}^+$, is the minimum number of patches will be examined

Fully classified patch dictionary $D\{d_{LR}, d_{HR}, d_{DIFF}\}$, for each sub-class with code $Code_{SD}$, the indexed sub-dictionary is $SD\{sd_{LR}, sd_{HR}, sd_{DIFF}, sd_{index}\}$ with size T_{SD}, for $SD \in \mathbb{Z}^+$
A non-empty array $\mathrm{E}_{LR}[3][3]$, and its Diff array $\mathrm{E}_{DIFF}[12]$
Output:
The similarity fuzzy search results: $\mathrm{E}_{1st}[\mathrm{L}_s][\mathrm{L}_s]$, $\mathrm{E}_{2nd}[\mathrm{L}_s][\mathrm{L}\ \ ]$, and $\mathrm{E}_{3rd}[\mathrm{L}_s][\mathrm{L}_s]$
Procedure:
(1) **Find** code $Code_{SD}$ for $\mathrm{E}_{DIFF}[12]$ and **locate** the sub-dictionary $SD\{sd_{LR}, sd_{HR}, sd_{DIFF}, sd_{index}\}$
(2) **Calculate** $dist_{zero}$ for $\mathrm{E}_{LR}[3][3]$ and $\mathrm{E}_{DIFF}[12]$
$dist_{part1} = distance_{DTW}(flat(\mathrm{E}_{LR}[3][3]), flat(zeros(\mathrm{E}_{LR}[3][3])))$
$dist_{part2} = distance_{DTW}(\mathrm{E}_{DIFF}[12], zeros(\mathrm{E}_{DIFF}[12]))$
$dist_{zero} = dist_{part2} + dist_{part2}$
(3) **Calculate** absolute differences between $dist_{zero}$ and each element in sd_{index}
(4) **Sort** sd_{index} based on the absolute differences in ascending order
(5) **Find** $dist_{thd}$ threshold as $dist_{thd} = sd_{index}[N_{patch}]$, **if** $(T_{SD} \leq N_{patch})$, **then** $\{dist_{thd} = sd_{index}[T_{SD}]\}$
(6) **For each** t **in** T_{SD} **do** {
If $(sd_{index}[\mathrm{t}] \leq dist_{thd}\ )$ **then** {
Let 1st_best_so_far = infinity
Let 2nd_best_so_far = infinity
Let 3rd_best_so_far = infinity
For each t **in** T_{SD} **do** {
$dist_{part1} = distance_{DTW}(flat(\mathrm{E}_{LR}), flat(sd_{LR}[t]))$
$dist_{part2} = distance_{DTW}(\mathrm{E}_{DIFF}, sd_{DIFF}[t])$
$dist_{all} = dist_{part2} + dist_{part2}$
If $(dist_{all} <$ 1st_best_so_far) **then** {
1st_best_so_far $= dist_{all}$
$\mathrm{E}_{1st} = sd_{HR}[t]$ }
Else if $(dist_{all} <$ 2nd_best_so_far) **then** {
2nd_best_so_far $= dist_{all}$
$\mathrm{E}_{2nd} = sd_{HR}[t]$ }
Else if $(dist_{all} <$ 3rd_best_so_far) **then** {
3rd_best_so_far $= dist_{all}$
$\mathrm{E}_{3rd} = sd_{HR}[t]$ }
Else {**Continue**}
}
}
Else {**Continue**}
}
(7) **Return** $\mathrm{E}_{1st}[\mathrm{L}_s][\mathrm{L}_s]$, $\mathrm{E}_{2nd}[\mathrm{L}_s][\mathrm{L}_s]$, and $\mathrm{E}_{3rd}[\mathrm{L}_s][\mathrm{L}_s]$

3.4 Implementation

The downscaling method was implemented using Python and open-source scientific libraries, including NumPy, NetCDF, numba, and itertools. The proposed downscaling method is implemented as a batch of computation processes from dictionary construction to downscaled output production. The system accepts NetCDF and HDF files as input and output downscaled results in NetCDF formats. The current implementation method is not under any existing frameworks (to maximize implementation flexibility), the implementation of dictionary learning is based on programming each algorithm in individual programs, and could be treated as a complete downscaling system.

3.5 Precipitation Downscaling Case Studies

3.5.1 General Design of Downscaling Experiments

3.5.1.1 Introduction

To examine the performance of the proposed downscaling method on the gridded precipitation product, two case studies have been conducted in this chapter. The first case study is developed as a synthetic experiment with direct ground truth available for comparison, and the second case study is a real-world use case in which no direct ground truth is available to compare with the downscaled results. The general design and detailed setups of the experiments are discussed in this section.

3.5.1.2 Downscaling Goal and Objective

The goal for this downscaling research is to downscale GCMs' outputs to the user's desired resolution, e.g. mesoscale (200km to 2km) or even microscale (less than 1km). However, different end-users would have different requirements for the final output. In this downscaling research, downscaling functional requirements and non-functional requirements are developed based on the current challenges discussed in the introduction chapter. Specifically, the ability to produce precipitation fields is highly valued in this study and yields a base requirement for all the downscaling methods that are investigated in this research.

3.5.1.2.1 Downscaling Functional Requirements

There are several downscaling requirements guiding the precipitation case studies in this research:

1) Precipitation field generation: the downscaled results should be in the form of continuous precipitation field without any gaps in space
2) Data independence: coarse-resolution gridded precipitation variable should be used as the only input for downscaling, other variables shall not be included to avoid data dependence when downscaling future estimation of local precipitation. Historical precipitation data could be used in modeling or training, excluding the input data.
3) The downscaling method should be able to produce hourly precipitation results. Daily, monthly or larger time frames are not included in this study scope.

3.5.1.2.2 Downscaling Non-Functional Requirements

In addition to the functional requirements, some non-functional requirements are also considered as important factors in downscaling, including downscaling speed, computational complexity, and resource consumption. They should be maintained at a usable level, e.g. if downscaling a small data set would require hours of computing time, then it could not be accepted as a valid approach. However, modeling time and training time could be excluded from this requirement as they are usually a one-time cost.

3.5.1.3 Synthetic Experiments and Real-World Use Cases

A common problem in evaluating the downscaling method is the lack of ground truth data. Climate variables like precipitation are simulated differently across models and show different patterns and rainfall amounts. So, instead of downscaling a GCM output directly, synthetic experiments are often used to evaluate downscaling performance. Synthetic experiments are conducted through upscaling higher resolution rainfall products and using them as inputs and comparing downscaled results to original products to evaluate overall performance (Rebora 2006; He 2016). The upscaling process is achieved through aggregate grids as averages from the higher resolution.

3.5.1.4 Study Areas

The main study area is the Chesapeake Bay, which is an estuary in the Eastern U.S. Additional areas are also included for parallel comparison of the downscaling results. The specific areas are separated regions within California, Kansas, and Florida. Downscaling methods are used for all four regions and more sets are generated for the Chesapeake Bay for long-term comparison. Figure 12 maps the geographical coverage of each study region on the U.S. map.

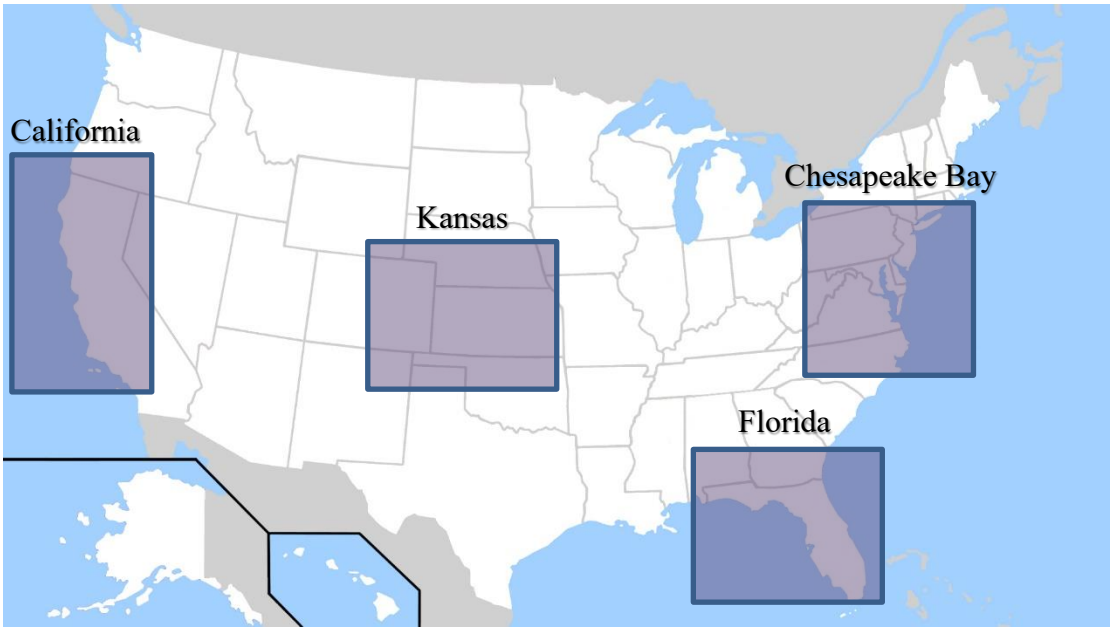


Figure 12 Study areas for downscaling studies

3.5.1.5 Datasets

Two datasets were used in the case studies, IMERG and MERRA-2. The first one is for the synthetic experiment and dictionary training and the second one tested as the real-world use case.

3.5.1.5.1 IMERG

The Integrated Multi-satellitE Retrievals for GPM (IMERG) dataset is collected and selected as experimental data which is generated by NASA's Precipitation Processing System every half hour with 4-hours (Early) to 3.5-months (Final) latency from acquisition time. IMERG V06 Final is chosen, which is a satellite-gauge product. The

spatiotemporal resolution of the precipitation in IMERG is 0.1° with half-hour reads for each day of the year.

3.5.1.5.2 MERRA-2

The MERRA-2 dataset is collected and selected as the experiment data, which is produced and provided by the Global Modeling and Assimilation Office of NASA Goddard Space Flight Center. MERRA-2 dataset is stored in NetCDF4 format and contains about 49 variables (e.g., Surface Wind Speed, Precipitation, Surface Air Temperature, etc.). PRECTOTCORR is chosen as the variable to use in this experiment, which is the bias-corrected precipitation output from an atmospheric model. The spatiotemporal resolution of this variable is 0.625° by 0.5° with hourly reads of the year.

3.5.1.5.3 Station Data (Gauge Data)

Station measurement of rainfall is an indirect way to measure precipitation, which could be highly biased depending on the local and gauge types. However, it is still the closest way to measure how much precipitation had fallen to the ground. The station data used in this research was collected from NOAA, National Climatic Data Center (NCDC) through Climate Data Online (CDO). Although the station data are the direct read from the rain gauges and provide hourly measurements, the recording time frame is different from station to station. For example, station A may start the measurement at the 47th min of each hour, and Station B always starts to measure at the 17th min of each hour.

3.5.1.6 Downscaling Methods

The methods used in downscaling case studies are chosen based on the ability to provide precipitation fields as the final output and whether they require climate variables

other than precipitation when producing downscaled results. Besides the statistical downscaling methods, bilinear interpolation (e.g., bicubic) is considered the baseline for the spatial downscaling of precipitation fields (He 2016). Specifically, three methods are selected to evaluate besides the proposed method: 1) bicubic interpolation, 2) DeepSD, as an SRCNN based super-resolution downscaling method (Vandal et al. 2017), and 3) RainFARM (Rebora 2006) as a stochastic downscaling method. Although there are other downscaling methods available in the research, most of them are not able to produce precipitation fields as output, require other datasets when downscaling, or they are limited to a specific area. Statistical methods like regression and PP methods are not chosen because they failed to produce continuous precipitation fields and require additional variables other than precipitation, which are not suitable for downscaling requirements in the previous section. Most of the statistical methods are involved using either observation data from the same time frame or other variables if precipitation field is desired as the downscaled result. Under this consideration, statistical methods are chosen too.

3.5.1.7 General Procedure

The downscaling process varies for different downscaling approaches. GCMs' precipitation product was the input, and the model or dictionary that was used in the production process could be trained or adjusted using historical datasets including those from other sources. However, the input itself or other variables were not used in the training process, i.e. the input was independent of the modeling procedure. A pre-test was utilized to test the performance of the downscaling method on a small amount of data. If

the results are under the baseline, then the corresponding downscaling method will not be tested against large sets or in other regions.

3.5.1.8 Validation and Evaluation

The validation and evaluation of precipitation downscaling are as difficult as downscaling itself. Evaluation is often case-by-case, clarifies the general criteria of a good result. The following items are generalized from literature and empirical studies.

3.5.1.8.1 Validation

When the ground truth is available, the downscaled results could be compared directly with it (as in the case study 1). Yet, data validation becomes a big problem when there is no easily obtained ground truth. Since precipitation measurements are different from product to product, and station level data could be highly biased; it rises a data validation challenge of the downscaled short-duration precipitation field. The data validation process is conducted by comparing a large number of the station reads with the downscaled results in case study 2 where there is no direct ground truth.

3.5.1.8.2 Evaluation

Although the evaluation of downscaling methods is often case-by-case, there are some general agreements in measuring the quality of the downscaled results. Specifically, the following items are generalized from literature and empirical studies which could estimate the quality of different downscaling methods:

1) *Large-scale rainfall amount consistency*: The downscaled product should be able to aggregate back to coarse resolution products to ensure the downscaled output is not another precipitation product (unbiased transformation).

2) *Small-scale rainfall amount prediction (statistically)*: The average value, maximum value, and standard deviation from the downscaled product should be close to the ground truth. RMSE should be as small as possible, and R^2 should be as close to 1 as possible. This part ensures the downscaled output values are not skewed.

3) *Small-scale spatial pattern prediction (visually)*: The general patterns, pattern edges, and pattern transactions should all be visually close to ground truth to ensure the downscaled output looks like a rainfall event.

4) *Other*: Data Dependency, Computing Intensity, Area limitation, etc.

Therefore, if the downscaled model satisfies those rules, it could be called a good downscaling model.

3.5.2 Case Study 1: Downscale Aggregated IMERG Precipitation

3.5.2.1 Introduction

This case study is a synthetic experiment using observation-based high-resolution precipitation gridded products.

3.5.2.2 Data

IMERG precipitation product was utilized in this case study. Specifically, the product is the V06 final Gauge Corrected dataset, which has a 3.5 months delay from the observation. The native resolution for IMERG is 0.1° longitude and 0.1° latitude, with half-hour observation. The year 2014 to 2018 are selected in the experiment, and 2014 to 2017 were used for any model or dictionary training process, while 2018 is used for method comparison in the production process.

3.5.2.3 Method

The downscaling ratio in this case study is 5, which is 5 times in longitude and 5 times in latitude, i.e. one grid point will be disaggregated to 25 points in space. Original IMERG data were aggregated 5 times to a coarse resolution, from 0.1° in both longitude and latitude to 0.5° in both longitude and latitude. Temporal aggregation is also executed to average half-hour reads to hourly reads. The spatiotemporal aggregated IMERG is used as inputs for different downscaling methods. Bicubic interpolation was used to establish a baseline for downscaling, DeepSD (Vandal et al. 2017) is selected as a representative of SRCNN based precipitation downscaling method, and RainFARM is selected as representative of the stochastic downscaling method. Historical IMERG data were used to train the model in DeepSD and the dictionary in the proposed method. A pre-test is introduced to give a basic view of the downscaling ability from different methods.

3.5.2.4 Results

Before applying all the methods, PreciPatch, DeepSD, and RainFARM were tested and compared with the bicubic interpolation method. According to the research result from (Vandal et al. 2017), DeepSD provided a downscaling result close to bicubic, which is not enough under this research's requirements.

3.5.2.4.1 Pre-test

As stated in the last section, bicubic interpolation is included as a downscaling method baseline. If methods are not working better than the bicubic method, then it is not a good idea to continue testing them against large datasets. Therefore, a small amount of

data was selected from a separate IMERG data set (IMERG without gauge correction) to test the general performance of the downscaling methods. DeepSD was trained using three months of IMERG data in the main study area (the Chesapeake Bay), and PreciPatch used the same training data set to construct the downscaling dictionary. RainFARM is a stochastic model, therefore no training is needed.

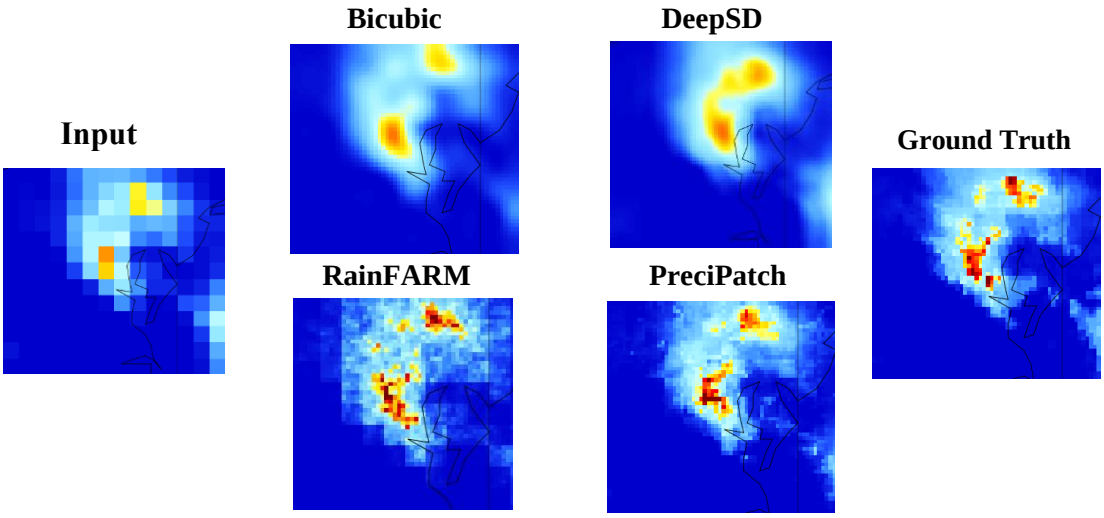


Figure 13 Downscaling pre-test results. The x-axis refers to the longitude, and the y-axis refers to the latitude. Snapshot of hourly precipitation at the Chesapeake Bay area on 2018/7/21 (23:30:00 UTC), downscaling from $\mathbf{0.5^{\circ}} \times \mathbf{0.5^{\circ}}$ to $\mathbf{0.1^{\circ}} \times \mathbf{0.1^{\circ}}$.

Figure 13 illustrates the pre-test results for DeepSD, RainFARM and the proposed method using the coarse-resolution IMERG as input. The result from DeepSD is very close to bicubic interpolation and simulated precipitation events as clustered clouds. RainFARM did a much better job than bicubic, which is closer to the ground truth. PreciPatch provides the best visual result in this pre-test, and the precipitation distributions were simulated close to the ground truth. Visually speaking, all three

methods are better than or close to bicubic interpolation, therefore they are all chosen for the next step in the experiment.

3.5.2.4.2 Experiment Results

For each study area, two images from different time slices are selected to show a brief view of the general performance of the bicubic method, DeepSD, RainFARM, and PreciPatch. The original high-resolution IMERG data set is also visualized to compare with the downscaled results from different methods as a direct visual comparison.

DeepSD is trained using the data from 2014 to 2017 at the same region, bicubic and RainFARM do not need a training process, and PreciPatch dictionary is trained using the whole U.S. coverage data from 2014 to 2017.

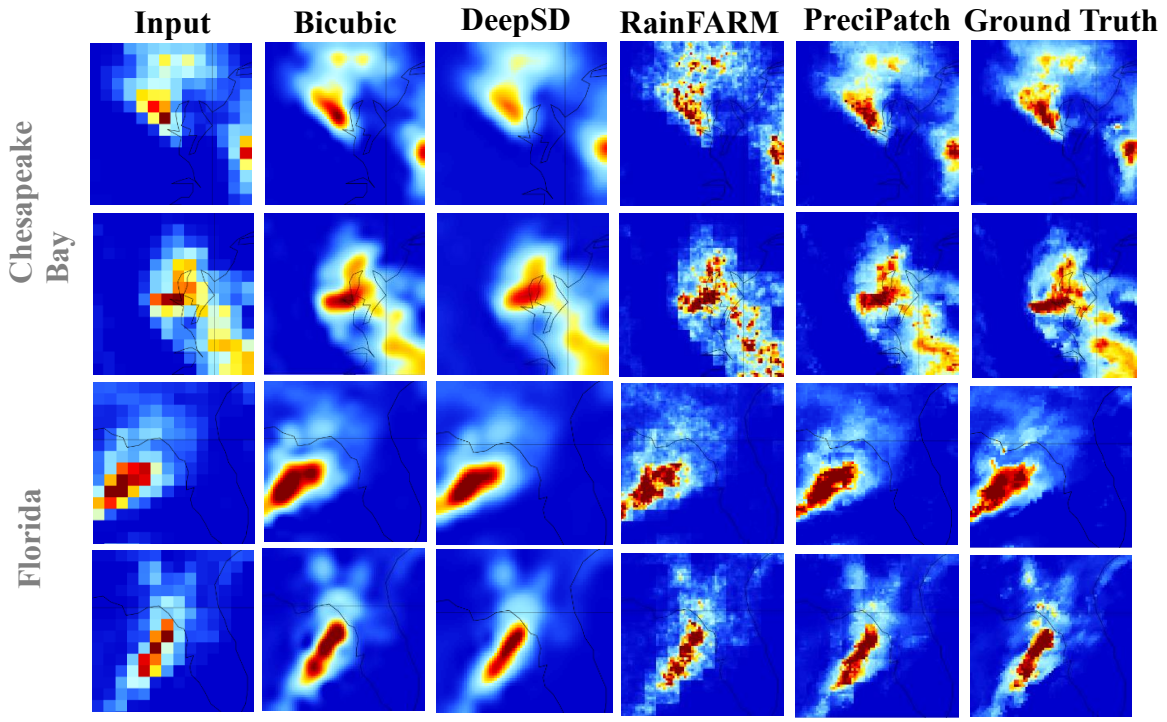


Figure 14 The Chesapeake Bay and Florida – IMERG. The x-axis refers to the longitude, and the y-axis refers to the latitude. Each row is the input aggregated Integrated Multi-satellitE Retrievals for GPM (IMERG) rain rate, Bicubic result, DeepSD result, RainFARM result, PreciPatch result and original high-resolution IMERG rain rate respectively for each location. 1st row: hourly precipitation at the Chesapeake Bay area on 2018/7/21 (23:30:00 UTC), 2nd row: hourly precipitation at the Chesapeake Bay area on 2018/7/21 (16:30:00 UTC), 3rd row: hourly precipitation at Florida area on 2018/12/14 (05:30:00 UTC), 4th row: hourly precipitation at Florida area on 2018/12/14 (18:30:00 UTC).

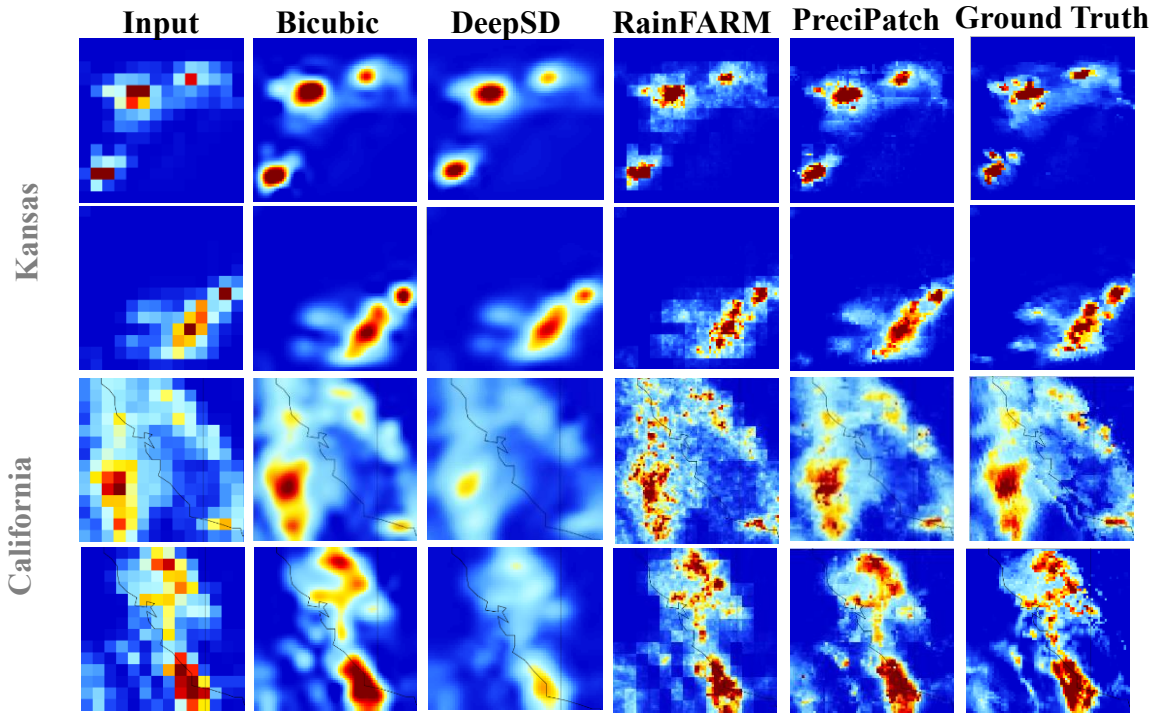


Figure 15 Kansas and California – IMERG. The x-axis refers to the longitude, and the y-axis refers to the latitude. Each row is the input aggregated Integrated Multi-satellitE Retrievals for GPM (IMERG) rain rate, Bicubic result, DeepSD result, RainFARM result, PreciPatch result and original high-resolution IMERG rain rate respectively for each location. 1st row: hourly precipitation at Kansas area on 2018/7/29 (04:30:00 UTC), 2nd row: hourly precipitation at Kansas area on 2018/7/29 (08:30:00 UTC), 3rd row: hourly precipitation at California area on 2018/1/9 (00:30:00 UTC), 4th row: hourly precipitation at California area on 2018/1/9 (08:30:00 UTC).

Figure 14 and 15 present the downscaled results for different study areas. The input column in each figure contains the inputs for different downscaling methods and is used as the only input in downscale methods. The results from bicubic, DeepSD, RainFARM, and PreciPatch are in the next four columns, respectively. Original IMERG data are in the last column for visual comparison with different results. Generally, PreciPatch outperformed RainFARM, DeepSD and bicubic, showing results closer to the ground truth in most cases. RainFARM method is a valid solution that meets downscaling requirements and can produce downscaling results with fast speed. However, due to theoretical limitations of this method which assumes rainfall fields as Gaussian random

fields at all scale levels (Rebora 2006), the practical shapes and textures of rainfall are not sufficiently simulated by RainFARM. Bicubic is often treated as the baseline for downscaling, and the results from DeepSD are very close to those from bicubic. DeepSD is showing great potential for improvements.

3.5.2.5 Validation and Evaluation

The downscaled results are compared with ground truth statistically.

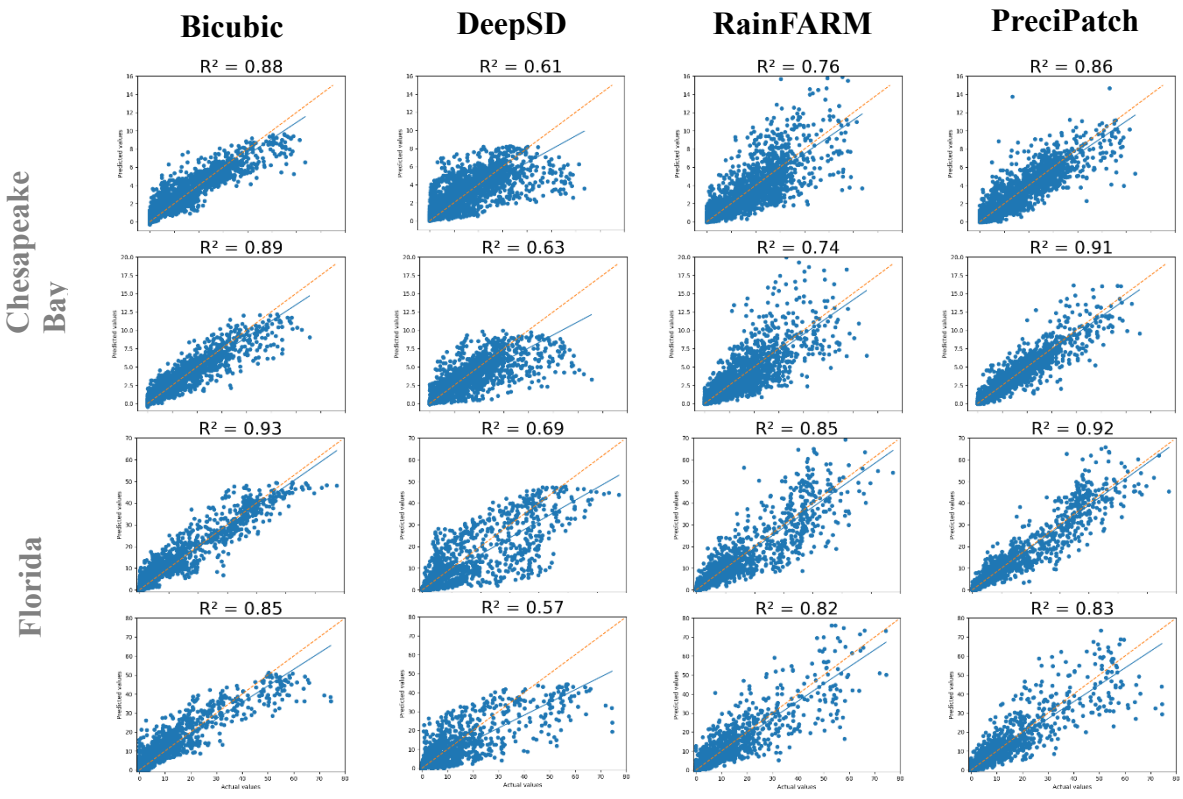


Figure 16 The Chesapeake Bay and Florida – R^2. The x-axis refers to the ground truth, and the y-axis is the prediction. The 45 degrees line in each graph is the perfect match reference line between the prediction and the ground truth, showing as the dashed line, the solid line is the trending line. 1st row: hourly precipitation at Chesapeake Bay area on 2018/7/21 (23:30:00 UTC), 2nd row: hourly precipitation at Chesapeake Bay area on 2018/7/21 (16:30:00 UTC), 3rd row: hourly precipitation at Florida area on 2018/12/14 (05:30:00 UTC), 4th row: hourly precipitation at Florida area on 2018/12/14 (18:30:00 UTC).

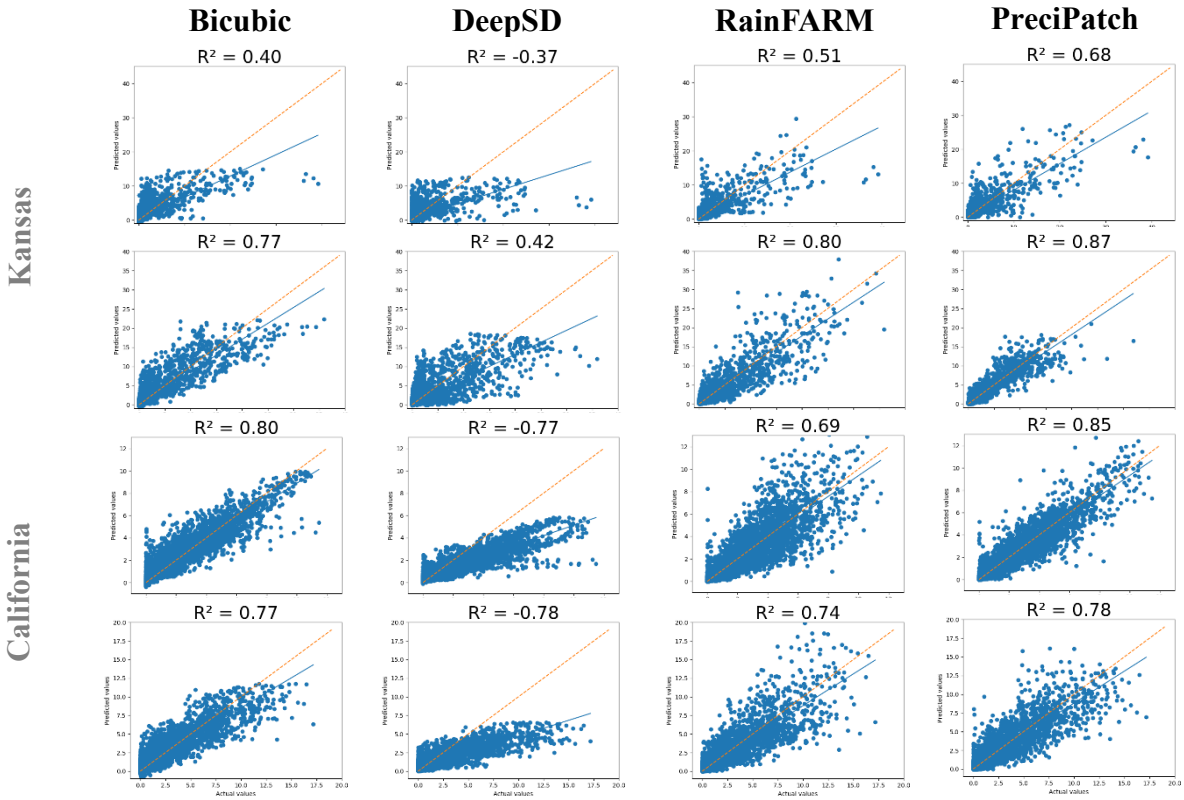


Figure 17 Kansas and California - R^2. The x-axis refers to the ground truth, and the y-axis is the prediction. The 45 degrees line in each graph is the perfect match reference line between the prediction and the ground truth, showing as the dashed line, the solid line is the trending line. 1st row: hourly precipitation at Kansas area on 2018/7/29 (04:30:00 UTC), 2nd row: hourly precipitation at Kansas area on 2018/7/29 (08:30:00 UTC), 3rd row: hourly precipitation at California area on 2018/1/9 (00:30:00 UTC), 4th row: hourly precipitation at California area on 2018/1/9 (08:30:00 UTC).

Figure 16 and 17 shows the direct comparison between the downscaled point values (predicted values) to the ground truth in the cases corresponding to the ones showing in Figure 14 and 15. Each plot represents a time slice. Statistically speaking, for R^2, bicubic and PreciPatch performs better than DeepSD and RainFARM in most cases. Six more criteria are used to evaluate the performance between different methods. A total of 2,208 time slices (hourly precipitation from 2018 June, July, and August) are used from all four study regions to generate results. The statistical comparison is made through

a grid-to-grid comparison between the predicted values from each method to the ground truth at each time slice. The detailed statistical comparison is shown in Table 13; the values are averaged from all time slices, where PreciPatch shows the best statistical accuracies in almost all aspects, followed by RainFARM and bicubic. DeepSD has comparatively low performance in this case study, which agrees with the results from the pre-test and visual comparison. PreciPatch outperforms RainFARM, bicubic, and DeepSD in this synthetic experiment regarding statistical accuracy. The R^2 core used in this research is conducted using sklearn and the negative values regarding R^2 suggest the model does not produce a good result in the prediction (Alexander, Tropsha, and Winkler 2015).

Table 13 Dictionary vs. RainFARM

mm/hour	Bicubic	DeepSD	RainFARM	PreciPatch
Average Bias per grid	0.03	0.12	4.3×10^{-9}	0.003
Average R^2	0.18	-0.16	0.29	0.42
Average (+) R^2	0.44	0.3	0.47	0.57
RMSE	0.45	0.61	0.49	0.42
Mean Absolute Error	0.13	0.28	0.13	0.1
Maximum Absolute Error	7.75	7.95	4.2	3.87
STDV Absolute Error	0.2	0.27	0.09	0.07

In addition to the visual and statistical comparison between methods, aggregation of the downscaled results was implemented to test the downscaling method's ability to hold precipitation consistency from large-scale to small-scale. As designed with constrains for both RainFARM and PreciPatch, the downscaled results from these two methods could be upscaled to the original coarse-resolution input with visually unnoticeable variations. However, the downscaled results from bicubic interpolation and DeepSD methods could not be aggregated back to low-resolution inputs, which means such downscaling approaches represent a biased transformation from large-scale data to small-scale outputs.

3.5.2.6 Indication of Results

The synthetic experiment conducted in this case study has tested the downscaling ability of the proposed method (PreciPatch), a classical stochastic method (RainFARM), and a newer adapted SRCNN based method (DeepSD). Unfortunately, DeepSD failed to produce a result that is visually better than the bicubic interpolation method in the pre-test. Alternatively, RainFARM, and PreciPatch both had good performances in the pre-test and real experiments. RainFARM can simulate precipitation fields at the local scale; however, the rainfall edges of the precipitation clusters are not well simulated. Meanwhile, the PreciPatch could simulate both the rainfall edges and distribution patterns closer to the ground truth.

3.5.3 Case Study 2: Downscale MERRA-2 Precipitation

3.5.3.1 Introduction

MERRA-2 dataset is a typical example of GCM's output. It is an output of a version of the GEOS-5 Atmospheric General Circulation Model (AGCM) (Molod et al. 2015). As a follow-on product of MERRA, MERRA-2 aims to provide long-term reanalysis to simulate the water and energy cycles (Feng and Wang 2019). Currently, MERRA-2 products are widely used in many models and algorithms. For example, it is used in the IMERG V06 production algorithm (Huffman et al. 2015) and some local impact studies involving interpolation of original MERRA-2 (Guyonnet et al. 2019). Downscaling MERRA-2 to desired spatial resolution is beneficial in many aspects and lead to potential usages in several fields. For example, the downscaled results could aid associated research to produce more detailed analysis or modeling with higher resolution configuration.

3.5.3.2 Data

The data collected for this case study is the standard version of MERRA-2, which consists of 49 variables (e.g., Surface Wind Speed, Precipitation, Surface Air Temperature, etc.), including two precipitation variables: PRECTOT and PRECTOTCORR, and the latter one is the gauge-corrected product. The focus of this case study is the downscaling PRECTOTCORR variable. PRECTOTCORR was treated as the input low-resolution for the downscaling process, and the target is high-resolution PRECTOTCORR variable. MERRA-2 from the year 2018 is select as the inputs for downscaling.

3.5.3.3 Method

The downscaling ratio in this case study is the same as in the previous study, which is 5 times in longitude and 5 times in latitude. The original spatial resolution of MERRA-2 is 0.625° longitude and 0.5° latitude. Similar to the experiment plan and steps in case study 1, several downscaling methods are chosen to downscale MERRA-2 precipitation, including Bicubic interpolation, RainFARM, and the proposed PreciPatch method. SRCNN based method like DeepSD was not able to downscale MERRA-2 without historical high-resolution data, thus unable to produce a result in this case study. The general downscaling procedure for each downscaling method is listed below:

1) Bicubic interpolation: use MERRA-2 at each time slice as input and apply the bicubic interpolation algorithm to each grid point using the nearest 4 points.
2) RainFARM: use MERRA-2 at each time slice as input to the RainFARM model and produce results for each time step.
3) PreciPatch: use MERRA-2 at each time slice as input to do similarity search based on the trained dictionary in case study 1, find a high-resolution patch for each grid cell in MERRA-2 and linearly align the patches to produce the mask for downscaling. Use MERRA-2 as constrains to the mask to produce the final downscaled result. Ideally, if reliable higher-resolution of MERRA-2 is available, the best practice is to train a dictionary with historical high-resolution data. However, the absence of this data caused the dictionary approach in this case study to use the dictionary constructed with IMERG patches, which is not the optimal choice but still reasonable (IMERG is based

on observation and therefore would provide a better estimation of distribution patterns of precipitation events at its scale).

3.5.3.4 Results

MERRA-2 precipitation was tested in the same four study areas, the main study area, the Chesapeake Bay, and the other three locations.

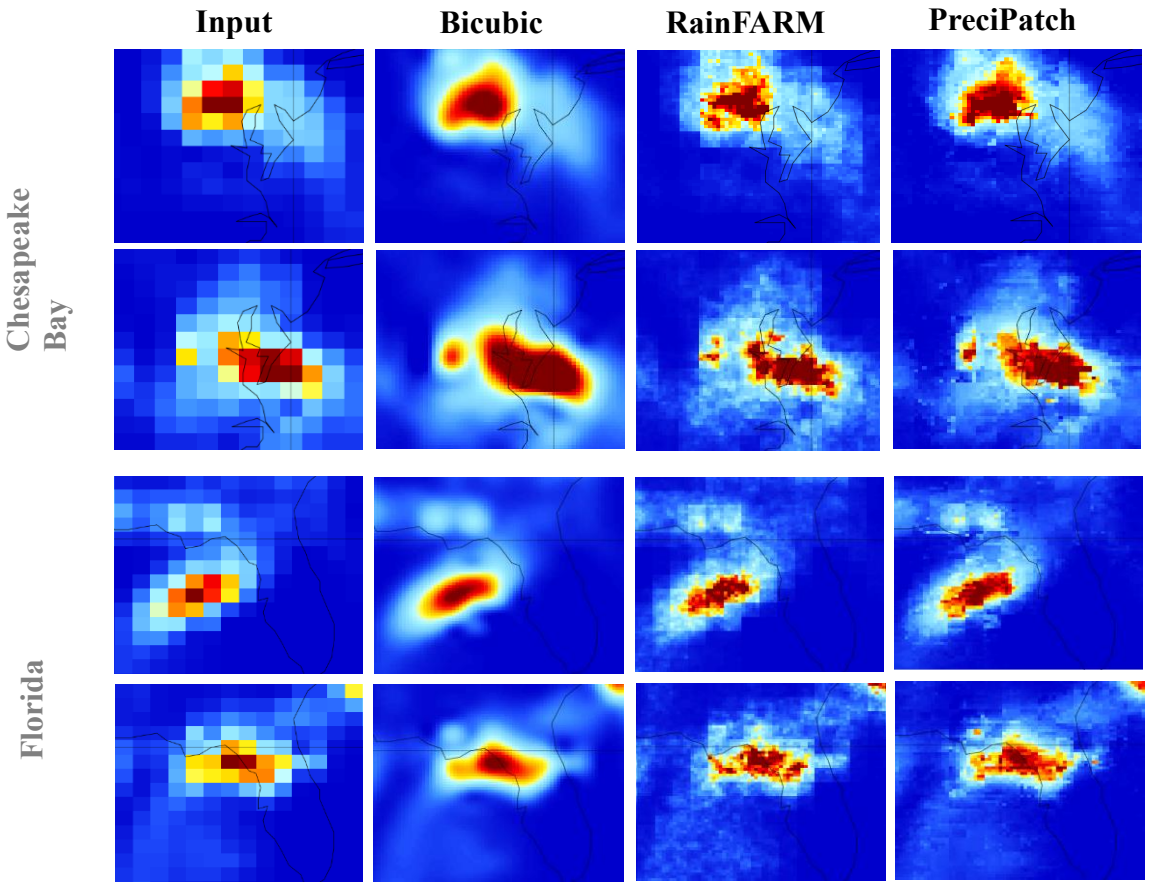


Figure 18 The Chesapeake Bay and Florida - MERRA2. The x-axis refers to the longitude, and the y-axis refers to the latitude. 1st row: hourly precipitation at the Chesapeake Bay area on 2018/7/21 (23:30:00 UTC), 2nd row: hourly precipitation at the Chesapeake Bay area on 2018/7/21 (16:30:00 UTC), 3rd row: hourly precipitation at Florida area on 2018/12/14 (05:30:00 UTC), 4th row: hourly precipitation at Florida area on 2018/12/14 (18:30:00 UTC).

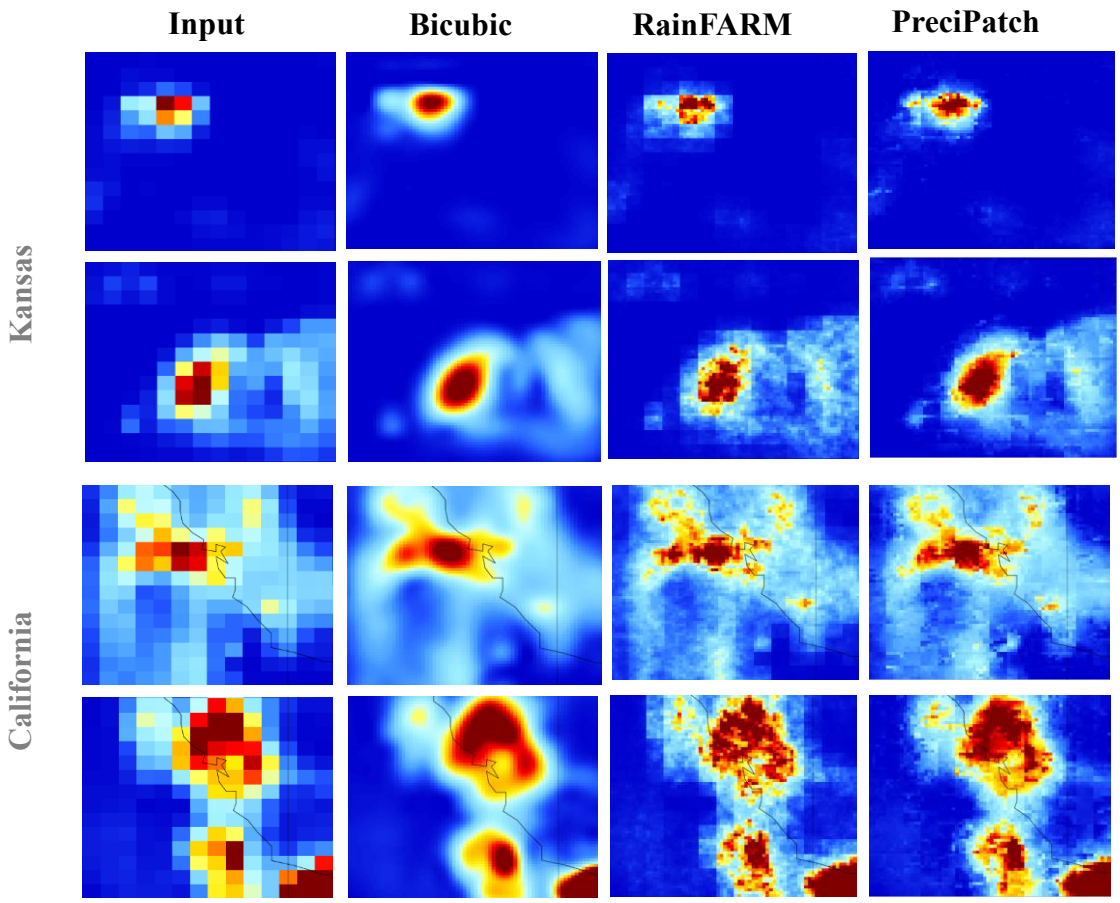


Figure 19 Kansas and California - MERRA2. The x-axis refers to the longitude, and the y-axis refers to the latitude. 1st row: hourly precipitation at Kansas area on 2018/7/29 (04:30:00 UTC), 2nd row: hourly precipitation at Kansas area on 2018/7/29 (08:30:00 UTC), 3rd row: hourly precipitation at California area on 2018/1/9 (00:30:00 UTC), 4th row: hourly precipitation at California area on 2018/1/9 (08:30:00 UTC).

Figure 18 to Figure 19 is the downscaling results for those areas. The coarse-resolution images (grid-based) in the input column are the inputs for different downscaling methods and are the only inputs. The next three columns contain the results from bicubic interpolation, the RainFARM method, and PreciPatch. Although there is no direct ground truth to compare, the visual representation of those results could give us a vague impression of their performance. The precipitation field simulated by the

RainFARM method is close to PreciPatch, but it lacks quality in simulated rainfall edges, which alternatively is well simulated by PreciPatch.

3.5.3.5 Validation and Evaluation

For the MERRA-2 downscaling case study, there is no high-resolution MERRA-2 precipitation gauge corrected data available for direct comparison with the downscaled results. Instead, station data was collected from 150 station sites in the main study area (the Chesapeake Bay) for the same time frame as the downscaled data (2018 June, July, and August). 2,208 time slices were chosen for comparison in total. After preprocessing and data cleaning, 201,250 records were considered as valid gauge reads and used in the data validation process. The closest grid point in the gridded datasets was chosen for each station point and the closest read (hourly reads) from the gridded dataset was indicated as the precipitation value to compare. In addition to the downscaled results from bicubic, PreciPatch, and RainFARM, the original MERRA-2 precipitation variable (gauge corrected) and original IMERG (V06 gauge corrected) were also compared with the raw station data to give a general idea of data error against station data. Table 14 lists detailed information for the comparison.

Table 14 Station data comparison

	Original MERRA-2	Bicubic	PreciPatch	RainFARM	Original IMERG
RMSE (mm/h)	2.555	2.672	2.565	2.568	2.657
Correlation	0.267	0.121	0.259	0.256	0.336

Absolute Bias per hour per station (mm/h)	0.659	0.636	0.662	0.664	0.0699

Across all the precipitation products and downscaled MERRA-2 results, they all set at a low level of data correlation with station data, which is expected by the authors. This low level of data correlation in hourly precipitation is due to the difficulty of obtaining accurate measurements of rainfall and inconsistency between precipitation models and observations in short-duration events. There are many known limitations of using station data for precipitation estimation, especially for hourly precipitation validation. Broadly speaking, the downscaled results' performance against station data is very close to its coarse-resolution input (MERRA-2), which meets the basic validation requirements for downscaling: data are not skewed or highly biased, the rainfall amount fits into the MERRA-2 estimation path. The downscaled results from PreciPatch and RainFARM could be accepted as valid results based on the limited information we can collect from stations. Although the comparison with station data could not be treated as the formal validation process, it does provide some comparisons between the downscaling methods when there is no ground truth available.

3.5.3.6 Multi-Dictionary Testing for General Coherence and Stability of Downscaled Results

The sparse coding used in the super-resolution field has proved the feasibility of using a fixed number of patches to estimate the missing information in a low-resolution to the high-resolution mapping process. However, utilizing the sparse coding as a

dictionary learning approach for precipitation downscaling has not yet be analyzed with real examples and experiments. This part of the section aims to provide examples for testing the general coherence and stability of PreciPatch when different data sets are ingested to construct the dictionary.

The datasets from downscaling case study 1 are used in this section for constructing different dictionaries. Specifically, U.S. coverage IMERG data was divided and subsetted into three regions: 1) the west region, 2) the central region, and 3) the east region. The data in each region were trained into separate dictionaries without overlapping. Figure 20 demonstrated the workflow of this process.

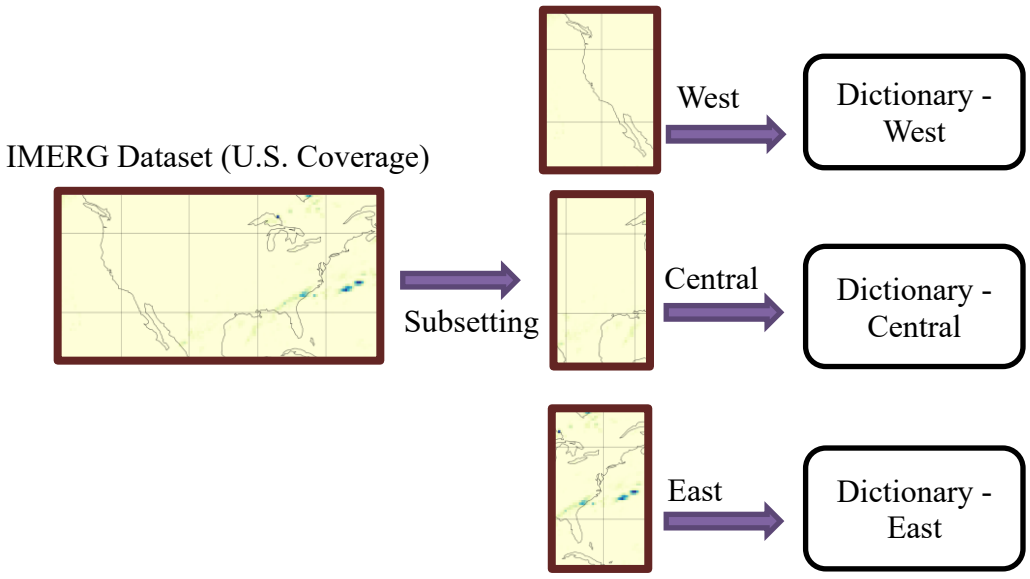


Figure 20 Multi-dictionary training

After classification and indexing each dictionary, MERRA-2 data are used as inputs to test the performance of each dictionary. The full dictionary which was trained by using the data from the whole U.S. and is also tested to produce comparable results.

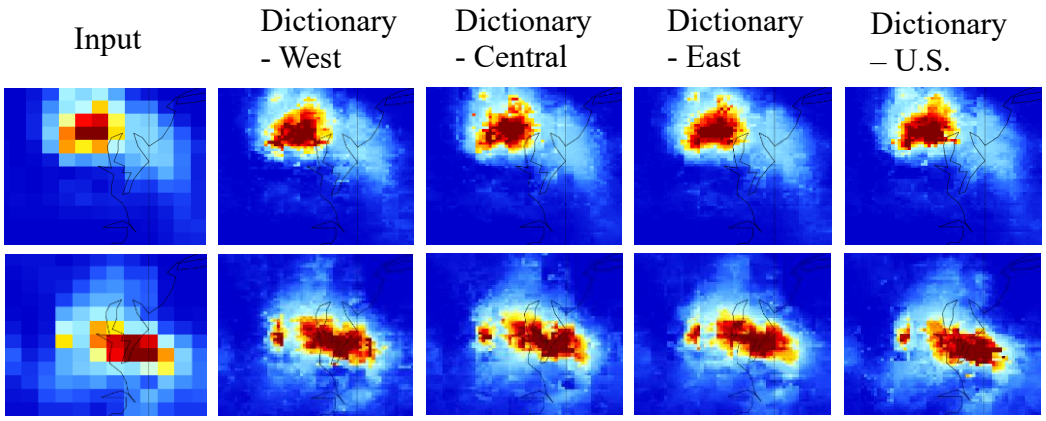


Figure 21 Downscaled results from different dictionaries. The x-axis refers to the longitude, and the y-axis refers to the latitude. 1st row: hourly precipitation at the Chesapeake Bay area on 2018/7/21 (23:30:00 UTC), 2nd row: hourly precipitation at the Chesapeake Bay area on 2018/7/21 (16:30:00 UTC).

Figure 21 presents the downscaled results by using different dictionaries on the same input. Two timeframes were selected to compare. The downscaled results are very close to each other as expected from sparse coding. Very few of the variations could be observed from the graphs and the full dictionary is showing the most visually appealing result. The results that were produced from different dictionaries indicate a general conference among the trained dictionaries and the stability of the dictionary learning method in a broad sense.

3.5.3.7 A Step Further

Downscaling precipitation events to mesoscale-gamma (2-20km) or even microscale (less than 2km) is the final goal of all downscaling studies related to local impact studies. However, it is an even harder challenge for downscaling methods based on prior knowledge. The stochastic downscaling method like RainFARM does not have this problem because it assumes the statistical distribution pattern at all scales following a Gaussian distribution pattern, which is not always the truth and has many limitations when applying.

One assumption is made for this downscaling test: precipitation distributions would follow the same pattern when the same downscaling ratio is applied. Although this statement is not true in every case, it does motivate further scale downscaling tryouts. A a better way of using PreciPatch is to find observation gridded data at a similar level with the target resolution. A quick implementation would be using the same dictionary that is used in the current MERRA-2 downscaling use case. The trained dictionary is treated as a 5 times enlargement lens in this test.

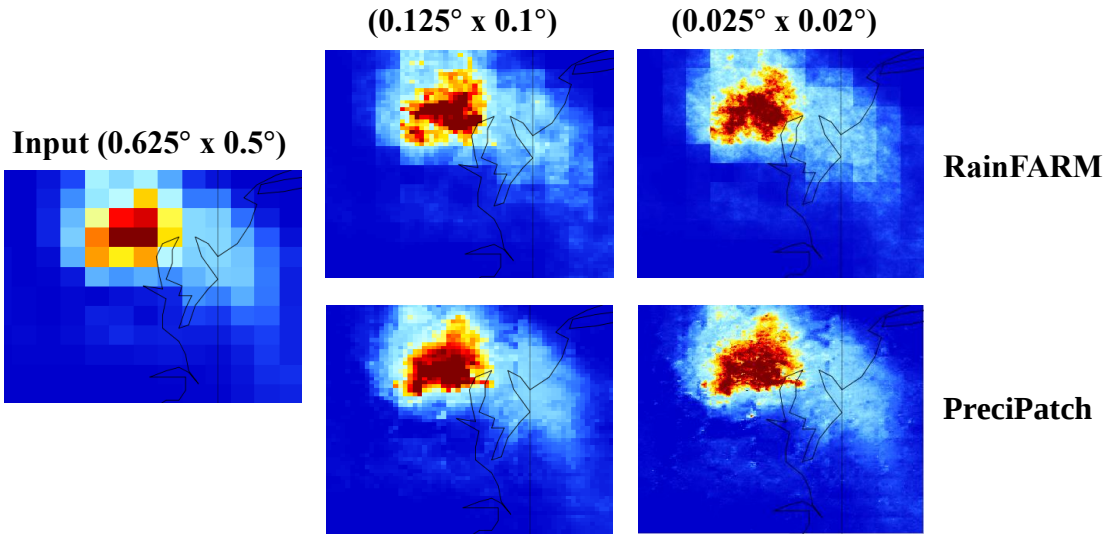


Figure 22 Further downscaling of MERRA2 precipitation. The x-axis refers to the longitude, and the y-axis refers to the latitude. 1st row: hourly precipitation at the Chesapeake Bay area on 2018/7/21 (23:30:00 UTC), 2nd row: hourly precipitation at the Chesapeake Bay area on 2018/7/21 (16:30:00 UTC).

Figure 22 displays the final results from this test. The downscaled results are visually appealing visually, and precipitation patterns were sharpened to the next level. The results indicate a potential for seeking downscaling methods at the microscale.

3.5.3.8 An Example of Full-Field Generation

An important criterion of evaluating precipitation downscaling method is the ability to provide full-field generation of precipitation fields. The traditional statistical downscaling methods are often criticized by their limited use when local observation data is not enough for a study region or even that no observation data are available (e.g. maintain glaciers). One of the approaches used in the statistical downscaling method is to migrate the regression mapping relationship from one area to another area (He 2016). Such approaches are not good practices because the linear or non-linear relationships found in a local area are often unique and not transferable.

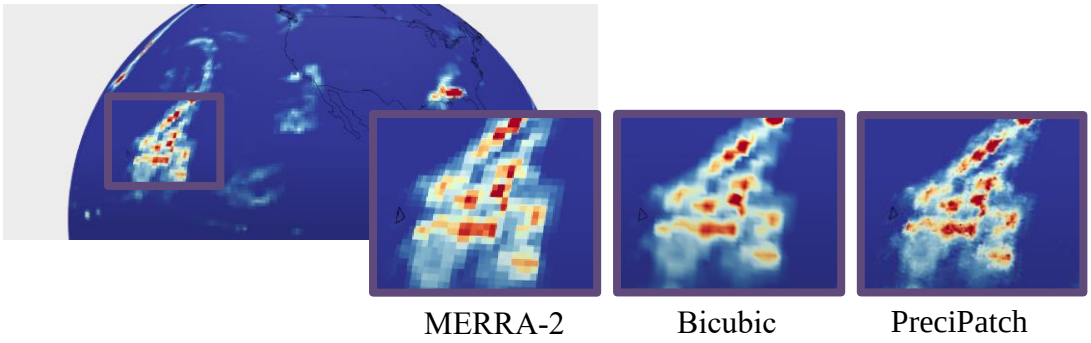


Figure 23 An example of full-field generation

Proposing a method that is capable of full-field generation is included in the design of the dynamic dictionary learning method and fits the concept of sparse coding. The trained dictionary can produce downscaled results regardless of the location of the input. Figure 23 gives a brief view of a downscaled global precipitation field. Compare to the bicubic interpolation, PreciPatch has a better simulation result for precipitation in the form of providing clear patterns in the precipitation cluster. However, when applying such an approach, different dictionaries trained using data from different locations is preferred as the possibility of finding a similar precipitation pattern could be dramatically increased if local historical data is utilized in the dictionary.

3.5.3.9 Indication of Results

This downscaling case study is showing the power of the proposed method when applying to real-world examples. The proposed method (PreciPatch) meets the functional requirements of precipitation downscaling (discussed in section 3.5.1.2.1 of this chapter).

Thus, it proved to be a valid downscaling method and can downscale hourly precipitation into precipitation fields. Due to the lack of accurate ground truth, comprehensive validation and evaluation could not be done to further justify the correctness of the downscaled results. The validation of PreciPatch applying to MERRA-2 would require further support from dynamic downscaling and the availability of high-resolution ground truth.

CHAPTER 4. CONCLUSION AND FUTURE WORK

The goal of this dissertation research is to contribute to geoscience and local climate impact studies with a big climate data storage method and a precipitation downscaling method. The two methods are to 1) seek high performance and cost-efficient solution for gridded climate data storage and query, and 2) investigate current downscaling methods and propose a new precipitation downscaling method for gridded precipitation data. Methods are proposed for each part of this dissertation research. Specifically, 1) an array database engine (LotDB) is developed based on a new N-Dimensional hash function and the integration with a unified storage structure and memory-mapping technology, and 2) a precipitation downscaling method (PreciPatch) based dynamic dictionary learning is proposed. Database comparison experiments and downscaling case studies have been used to evaluate the proposed methods' abilities in solving current big climate data storage and access challenges, and produce higher resolution climate indicators for local-scale impact study challenges.

4.1 Conclusion

4.1.1 An Array Database Engine (LotDB) for Climate Gridded Datasets

Although array is one of the oldest data structures, the study of storing and retrieving large multidimensional array datasets are limited. Earth observations and climate model simulations are producing larger amounts of output in multidimensional

array format due to increases in model resolution and remote sensing technologies. However, it is challenging to provide solutions to handle big multidimensional datasets. I designed and implemented a solution for efficient gridded data storage and faster data retrieval while being cost-effective. I reviewed past research for storing multidimensional arrays in both relational databases and array databases. Array databases are more native and advanced solutions than relational databases. Current solutions still have their limitations regarding query performance and data volume expansion. Therefore, an n-dimensional hash function algorithm was proposed to perform a fast data retrieval action on a unified data storage structure, and a prototype database library (LotDB) was developed by integrating memory-mapping technology and this algorithm. PostgreSQL, MongoDB, and SciDB were selected to compare the performance with LotDB using MERRA-2 data storage and retrieval. The preliminary experimental results have shown promising potentials of LotDB for efficient multi-dimensional gridded climate data storage, and abilities for fast data retrieval. The run-time results are validated by using multiple timers and repeating the same experiments several times. Also, to avoid the effect of using in-memory cache in comparison, physical memory is cleaned each time before the query execution and cold-run time is recorded instead of the hot-run. Therefore, the results are credible for general analysis. Yet, the standalone mode for NoSQL databases are far less potent than the clustered mode, MongoDB and SciDB would have better results if they were deployed in a cluster.

Main differences between this dissertation and previous array database research include:

1) Utilizing the unified storage structure instead of chunked storage
2) A decentralized index system vs. a centralized index system in other databases
3) Integrating memory-mapping technology into array database studies

Even with the upsides, there are some limitations to this research. For example, empty cells in a gridded climate dataset will count as full stored cells with value "NaN," which is not a problem for ordinary datasets but a waste of storage space for sparse matrix. This part of the dissertation research makes contributions to climate data management studies and climate change study in general in the form of offering an N-Dimensional hash function for a fast query on array-based climate data and a database engine as a quick implementation, in detail:

1) This method could improve the general performance of gridded data analysis by reducing the data storage cost and access latency.
2) Since a large portion of climate data is gridded data, this research provides additional options for cost-efficient and high-performance solutions on handling gridded data.

4.1.2 A Gridded Precipitation Downscaling Method (PreciPatch)

The importance of precipitation downscaling is well recognized in hydrological impact studies and local climate change-related domains. However, few statistical downscaling methods can provide universally applicable downscaling solutions for arbitrary GCM outputs, large amounts of precipitation downscaling related studies are area limited and highly additional data-dependent, not applicable for random locations. Meanwhile, major research can only provide the downscaled result on a level of single or

multiple stations. Providing precipitation fields as outputs are still beyond the reach for most traditional methods. Recent developments in the stochastic downscaling method and the SR based method brought new opportunities to enhance downscaling studies. A new precipitation downscaling method based on dynamic dictionary learning is proposed with detailed algorithms and procedures. Two precipitation downscaling study cases are conducted to evaluate the performance of the proposed method and compare it with other downscaling methods (RainFARM and DeepSD), along with the comparison baseline, the bicubic interpolation method. Results from case studies have shown positive feedback on the proposed downscaling method, demonstrating good quality for simulating spatially distributed precipitation fields and yielding the best performance among all other tested methods through visual inspection and quantitative analyses. It is hard for traditional and machine learning super-resolution based methods to exclude the interpolation process and include the non-linear characteristics of precipitation event into the method design (Yu et al. 2020). Meanwhile, many downscaling methods like bicubic interpolation and DeepSD do not have extra constraints to enforce the final outputs to hold rainfall consistency with the initial inputs. The proposed method interprets the downscaling task as a constrained single image super-resolution problem, different from traditional approaches and other downscaling methods, it addresses the precipitation downscaling issue using a novel approach which excludes the interpolation process and is able to generate non-linear spatial patterns that are closer to rainfall observations. Additionally, rainfall consistency is ensured by adding extra constraints in the proposed method. As a result, PreciPatch achieved better results than other downscaling methods in precipitation downscaling

study cases because it takes the spatial characteristics of precipitation into the method design consideration and optimizes the method to fit precisely for the precipitation downscaling scenario.

Precipitation field downscaling is a challenging task and related research topics have been explored by many studies. Major differences between this dissertation work and previous precipitation downscaling research include:

1) Full-field generation instead of area limited generation or fixed size.
2) Precipitation fields as target production instead of station networks (e.g., single or multiple), providing continuous information in 2D space.
3) Precipitation field distribution simulated closer to precipitation observation.
4) Data independence in production (available for downscaling future predictions)
5) Hourly precipitation downscaling to precipitation fields (downscaling short-duration precipitation events) instead of a daily average or seasonal downscaling in other researches.
6) Using the prior knowledge learned in one data set and successfully applied to another related data set.
7) The rainfall amount is consistent with large-scale vs. rainfall inconsistency in others.

Major differences between PreciPatch and other downscaling approaches regarding methodology includes: 1) comparing to statistical downscaling method: there is no need for other climate variables in the downscaling process, no regression process or probability assumption process, 2) comparing to stochastic downscaling method: the

proposed method is making use of prior knowledge, not purely hypothesis and mathematical transformation based, 3) comparing to SRCNN based methods like DeepSD: when interpolation process is involved, the final output is constrained by coarse-resolution inputs, which is equal to an unbiased disaggregation of precipitation amounting to space, 4) comparing to traditional dictionary learning: one master dictionary is constructed instead of two in traditional approaches, the proposed method uses dynamic patches to recover the high-resolution image, while the traditional approach uses stored fixed patches, DTW distance is used instead of MSE or Euclidean distance, fuzzy search is utilized in the new method instead of exact search for similarity patches, and rainfall amount is consistent in the downscaled result, 5) comparing to Analog Method (AM): similarity searching is executed spatially with a fuzzy search instead of linearly with a direct search.

In this research, there are also some limitations to PreciPatch. For example, the computing time is longer than other downscaling methods, which makes it hard to apply to large datasets and solve time-sensitive problems. The detailed comparison is in Table 15.

Table 15 Comparison of tested downscaling methods

	Bicubic	DeepSD	RainFARM	PreciPatch
Training time	N/A	~ 1 hour for 3 months of data (national coverage)	N/A	~ 200 hours for 300 Million patches (national

				coverage)
Applicability to different areas?	Direct apply, no training is needed	Additional training is needed if the area is different or the input size is different	Direct apply, no training is needed	Direct apply, no need for additional training
Computing time	< 0.001 sec per grid point	~0.005 sec per grid point	~0.005 sec per grid point	~2 sec per grid point per process
Overall downscaling performance	Baseline	Not good – close to the baseline	Rainfall clusters are not well simulated	Best
Holds rainfall consistency with large-scale input?	No	No	Yes	Yes

However, this limitation could be resolved by including an additional index layer for the dictionary and reprogramming the system using more efficient languages such as C/C++. Another limitation for the proposed method is its assumption: future precipitation would behave similarly as its past, relationships developed for present climate also hold for possible future climates (IPCC n.d.) (the same assumption is used in all statistical downscaling methods). One advantage compares with other models like DeepSD is that PreciPatch does not need additional training when applying to different areas, even for regions outside of the training area. The proposed precipitation downscaling method, PreciPatch has shown its ability to spatially downscale short-duration precipitation events

to precipitation fields and can be applied to downscale future estimations from GCMs or other gridded data sources. The bias correction is not addressed by PreciPatch, and the bias from GCMs are preserved at the same spatial scale as the input. This research contributes to hydrological impact studies and local climate change studies by providing a precipitation downscaling method and associated algorithms for simulating precipitation fields in local scale and being consistent with large-scale information, in detail:

1) This method does not require additional data as predictors to produce downscaled results, and it could be used for downscaling future estimations from GCMs.
2) This downscaling method has the potential to aid climate models like WRF by providing higher-resolution inputs.
3) The downscaled results from this method could be used to force mountain glacier models for local impact studies, where the complete absence of climate monitoring activities within the regions of interest presents a data challenge.

In general, the purpose of this downscaling research is not creating a new precipitation product for a region. It assumes that if there exists a high-resolution product of a low-resolution product, which follows the assumption that the high-resolution product holds grid-level rainfall consistency with the low-resolution product. PreciPatch will provide a good prediction of the high-resolution product using the low-resolution product as the input.

4.2 Future Works

The work presented in this dissertation shows the potential for seeking fast gridded data retrieval and efficient storage solutions using existing technologies. However, it is challenging to provide up-to-date solutions as the data size is also growing at an increasing speed. There are many directions for related future works, include but not limited to, 1) design a strategy for big data store and retrieval, 2) design and develop LotDB into a complete database system, and 3) extend current storage structure and algorithm to a distributed system. The current version of LotDB acts as a quick implementation of the N-Dimensional hash function algorithm, further implementations could be developed to enhance the data retrieval performance in large multidimensional arrays in different scenarios. For gridded precipitation downscaling, future works include: 1) adding more (spatiotemporal) indexing techniques to speedup the similarity search process in PreciPatch, 2) integrating with LotDB to further accerlerate the system, 3) converting and rebuilding PreciPatch on top of a machine learning framework to generalize the dictionary into a model and accelerate the downscaling process while reducing storage space of the dictionary, and 4) use different observation data to evaluate the downscaling method regarding expandability and transferability.

REFERENCES

Alexander, D.L., Tropsha, A. and Winkler, D.A., 2015. Beware of R 2: simple, unambiguous assessment of the prediction accuracy of QSAR and QSPR models. *Journal of chemical information and modeling*, 55(7), pp.1316-1322.

Ameri, P., Grabowski, U., Meyer, J. and Streit, A., 2014, September. On the application and performance of MongoDB for climate satellite data. In *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications* (pp. 652-659). IEEE.

Amirian, P., Basiri, A. and Winstanley, A., 2014, June. Evaluation of data management systems for geospatial big data. In *International Conference on Computational Science and Its Applications* (pp. 678-690). Springer, Cham.

Aniceto, R., Xavier, R., Holanda, M., Walter, M.E. and Lifschitz, S., 2014, November. Genomic data persistency on a NoSQL database system. In *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (pp. 8-14). IEEE.

Appel, M., Lahn, F., Buytaert, W. and Pebesma, E., 2018. Open and scalable analytics of large Earth observation datasets: From scenes to multidimensional arrays using SciDB and GDAL. *ISPRS journal of photogrammetry and remote sensing*, 138, pp.47-56.

Appel, M., Lahn, F., Pebesma, E., Buytaert, W. and Moulds, S., 2016, April. Scalable Earth-observation analytics for geoscientists: Spacetime extensions to the array database SciDB. In *EGU General Assembly Conference Abstracts* (Vol. 18). [online] Available at: https://ui.adsabs.harvard.edu/abs/2016EGUGA..1811780A/abstract [Accessed 12 Apr. 2019].

Archive.ipcc.ch., n.d. [online] Available at: https://archive.ipcc.ch/ipccreports/tar/wg1/380.htm [Accessed 4 May 2019].

Arritt, R.W. and Rummukainen, M., 2011. Challenges in regional-scale climate modeling. *Bulletin of the American Meteorological Society*, *92*(3), pp.365-368.

Atkinson, P.M., 2013. Downscaling in remote sensing. *International Journal of Applied Earth Observation and Geoinformation*, *22*, pp.106-114.

Barrodale Computing Services Ltd., 2002, Storing and Manipulating Gridded Data in Databases. [online] Available at: https://www-356.ibm.com/partnerworld/gsd/showimage.do?id=25877 [Accessed 11 Aug. 2019].

Baumann, P. and Stamerjohanns, H., 2014. Towards a systematic benchmark for array database systems. In *Specifying Big Data Benchmarks* (pp. 94-102). Springer, Berlin, Heidelberg.

Baumann, P., 1999, July. A database array algebra for spatio-temporal data and beyond. In *International Workshop on Next Generation Information Technologies and Systems* (pp. 76-93). Springer, Berlin, Heidelberg.

Baumann, P., Dehmel, A., Furtado, P., Ritsch, R. and Widmann, N., 1998, June. The multidimensional database system RasDaMan. In *Acm Sigmod Record* (Vol. 27, No. 2, pp. 575-577). ACM.

Baumann, P., Dehmel, A., Furtado, P., Ritsch, R. and Widmann, N., 1999, September. Spatio-temporal retrieval with RasDaMan. In *VLDB* (pp. 746-749).

Baumann, P., Dumitru, A.M. and Merticariu, V., 2013, August. The array database that is not a database: file based array query answering in rasdaman. In *International Symposium on Spatial and Temporal Databases* (pp. 478-483). Springer, Berlin, Heidelberg.

Baumann, P., Misev, D., Merticariu, V. and Huu, B.P., 2019. Datacubes: Towards Space/Time Analysis-Ready Data. In *Service-Oriented Mapping* (pp. 269-299). Springer, Cham.

Baumann, P., Misev, D., Merticariu, V., Huu, B.P., Bell, B., Kuo, K.S. and Bayesics, L.L.C., 2018. Array Databases: Concepts, Standards, Implementations. *Research Data Alliance (RDA) Working Group Report*. [online] Available at: https://www.rd-alliance.org/system/files/Array-Databases_final-report.pdf [Accessed 9 Sep. 2019].

Benestad, R., 2016. Downscaling Climate Information. Oxford Research Encyclopedia of Climate Science. [online] Available at: https://oxfordre.com/climatescience/view/10.1093/acrefore/9780190228620.001.0001/acrefore-9780190228620-e-27 [Accessed 14 Aug. 2019]

Benestad, R.E., Chen, D., Mezghani, A., Fan, L. and Parding, K., 2015. On using principal components to represent stations in empirical–statistical downscaling. *Tellus A: Dynamic Meteorology and Oceanography*, *67*(1), p.28326.

Bermowitz, R.J., 1975. An application of model output statistics to forecasting quantitative precipitation. *Monthly Weather Review*, *103*(2), pp.149-153.

Blumenfeld, J., 2015. Getting Petabytes to People: How the EOSDIS Facilitates Earth Observing Data Discovery and Use| Earthdata. *NASA, NASA*, *27*. [online] Earthdata.nasa.gov

Boé, J., Terray, L., Habets, F. and Martin, E., 2006. A simple statistical-dynamical downscaling scheme based on weather types and conditional resampling. *Journal of Geophysical Research: Atmospheres*, 111(D23).

Boncz, P.A., Kersten, M.L. and Manegold, S., 2008. Breaking the memory wall in MonetDB. *Communications of the ACM*, 51(12), pp.77-85.

Boncz, P.A., Zukowski, M. and Nes, N., 2005, January. MonetDB/X100: Hyper-Pipelining Query Execution. In *Cidr*(Vol. 5, pp. 225-237).

Brigadier, L., Allan, D., Noel, B., Luo, W., Chilekana, N. and Nyasa, L., 2016. Predictor selection associated with statistical downscaling of precipitation over Zambia. *Asian Journal of Physical and Chemical Sciences*, pp.1-9.

Bronstert, A., Kolokotronis, V., Schwandt, D. and Straub, H., 2007. Comparison and evaluation of regional climate scenarios for hydrological impact analysis: General scheme and application example. *International Journal of Climatology: A Journal of the Royal Meteorological Society, 27*(12), pp.1579-1594.

Brown, P.G., 2010, June. Overview of SciDB: large scale array storage, processing and analysis. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data* (pp. 963-968). ACM.

Busuioc, A., Von Storch, H. and Schnur, R., 1999. Verification of GCM-generated regional seasonal precipitation for current climate and of statistical downscaling estimates under changing climate conditions. *Journal of Climate*, *12*(1), pp.258-272.

Cannon, A.J., 2011. Quantile regression neural networks: Implementation in R and application to precipitation downscaling. *Computers & geosciences*, *37*(9), pp.1277-1284.

Carvalho, D., Rocha, A., Gómez-Gesteira, M. and Santos, C.S., 2014. WRF wind simulation and wind energy production estimates forced by different reanalyses: Comparison with observed data for Portugal. *Applied Energy*, 117, pp.116-126.

Chandler, W., Hoell, J., Westberg, D., Whitlock, C., Zhang, T. and Stackhouse, P. (2011). Downscaling NASA Climatological Data to Produce Detailed Climate Zone Maps. In: *American Solar Energy Society National Solar Conference*. [online] Available at: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20110011613.pdf [Accessed 14 Aug. 2019]

Chaney, N.W., Sheffield, J., Villarini, G. and Wood, E.F., 2014. Development of a high-resolution gridded daily meteorological dataset over sub-Saharan Africa: spatial analysis of trends in climate extremes. *Journal of Climate*, 27(15), pp.5815-5835.

Charles, S.P., Bates, B.C. and Hughes, J.P., 1999. A spatiotemporal model for downscaling precipitation occurrence and amounts. *Journal of Geophysical Research: Atmospheres*, *104*(D24), pp.31657-31669.

Charles, S.P., Bates, B.C., Whetton, P.H. and Hughes, J.P., 1999. Validation of downscaling models for changed climate conditions: case study of southwestern Australia. *Climate Research*, *12*(1), pp.1-14.

Chen, F., Liu, Y., Liu, Q. and Li, X., 2014. Spatial downscaling of TRMM 3B43 precipitation considering spatial heterogeneity. *International journal of remote sensing*, *35*(9), pp.3074-3093.

Chen, J., Brissette, F.P. and Leconte, R., 2012. Coupling statistical and dynamical methods for spatial downscaling of precipitation. *Climatic change*, *114*(3-4), pp.509-526.

Chen, J., Brissette, F.P. and Leconte, R., 2012. Downscaling of weather generator parameters to quantify hydrological impacts of climate change. *Climate Research*, 51(3), pp.185-200.

Chen, J., Brissette, F.P. and Leconte, R., 2014. Assessing regression-based statistical approaches for downscaling precipitation over North America. *Hydrological processes*, 28(9), pp.3482-3504.

Chen, J., Chen, H. and Guo, S., 2018. Multi-site precipitation downscaling using a stochastic weather generator. *Climate dynamics*, 50(5-6), pp.1975-1992.

Climate.gov., n.d. Climate Models | NOAA Climate.gov. [online] Available at: https://www.climate.gov/maps-data/primer/climate-models [Accessed 24 Aug. 2019]

Clune, T., Das, K., Duffy, D., Habermann, T., Huang, T., Kuo, K.S., Mattman, C. and Yang, C.P., 2015. Evaluation of Big Data Containers for Popular Storage, Retrieval, and Computation Primitives in Earth Science Analysis. [online] Available at: https://pdfs.semanticscholar.org/a3de/0cdb9bdd3c2d987e825ac774476a1e7a6c52.pdf [Accessed 19 Aug. 2019]

Conway, D. and Jones, P.D., 1998. The use of weather types and air flow indices for GCM downscaling. *Journal of Hydrology*, 212, pp.348-361.

Cornacchia, R., Héman, S., Zukowski, M., Vries, A.P. and Boncz, P., 2008. Flexible and efficient IR using array databases. *The VLDB Journal—The International Journal on Very Large Data Bases*, 17(1), pp.151-168.

Corti, P., Kraft, T.J., Mather, S.V. and Park, B., 2014. *PostGIS Cookbook.* Birmingham: Packt Publishing.

Coulibaly, P., Dibike, Y.B. and Anctil, F., 2005. Downscaling precipitation and temperature with temporal neural networks. *Journal of Hydrometeorology*, *6*(4), pp.483-496.

Cressie, N. and Wikle, C.K., 2015. *Statistics for spatio-temporal data*. John Wiley & Sons.

Cudre-Mauroux, P., Kimura, H., Lim, K.T., Rogers, J., Madden, S., Stonebraker, M., Zdonik, S.B. and Brown, P.G., 2010. Ss-db: A standard science dbms benchmark. Under submission. [online] Available at: https://www-conf.slac.stanford.edu/xldb10/docs/ssdb_benchmark.pdf [Accessed 11 Aug. 2019]

Cudré-Mauroux, P., Kimura, H., Lim, K.T., Rogers, J., Simakov, R., Soroush, E., Velikhov, P., Wang, D.L., Balazinska, M., Becla, J. and DeWitt, D., 2009. A demonstration of SciDB: a science-oriented DBMS. Proceedings of the VLDB Endowment, 2(2), pp.1534-1537.

Cuzzocrea, A., Song, I.Y. and Davis, K.C., 2011, October. Analytics over large-scale multidimensional data: the big data revolution!. In Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP (pp. 101-104). ACM.

d'Orazio, L. and Bimonte, S., 2010, September. Multidimensional arrays for warehousing data on clouds. In *International Conference on Data Management in Grid and P2P Systems* (pp. 26-37). Springer, Berlin, Heidelberg.

Dixon, K.W., Harris, L.M. and Knutson, T., n.d. Climate Model Downscaling – Geophysical Fluid Dynamics Laboratory. [online] Available at: https://www.gfdl.noaa.gov/climate-model-downscaling/ [Accessed 8 Jul. 2019].

Dong, C., Loy, C.C., He, K. and Tang, X., 2014, September. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision* (pp. 184-199). Springer, Cham.

Ebtehaj, A.M., Foufoula-Georgiou, E. and Lerman, G., 2012. Sparse regularization for precipitation downscaling. *Journal of Geophysical Research: Atmospheres*, *117*(D8).

Eden, J.M., Widmann, M., Grawe, D. and Rast, S., 2012. Skill, correction, and downscaling of GCM-simulated precipitation. *Journal of Climate*, *25*(11), pp.3970-3984.

Fatichi, S., Ivanov, V.Y. and Caporali, E., 2013. Assessment of a stochastic downscaling methodology in generating an ensemble of hourly future climate time series. *Climate Dynamics*, *40*(7-8), pp.1841-1861.

Fealy, R. and Sweeney, J., 2007. Statistical downscaling of precipitation for a selection of sites in Ireland employing a generalised linear modelling approach. *International Journal of Climatology: A Journal of the Royal Meteorological Society*, *27*(15), pp.2083-2094.

Feng, F. and Wang, K., 2019. Does the modern-era retrospective analysis for research and applications-2 aerosol reanalysis introduce an improvement in the simulation of surface solar radiation over China?. International Journal of Climatology, 39(3), pp.1305-1318.

Field, C.B. ed., 2014. *Climate change 2014–Impacts, adaptation and vulnerability: Regional aspects*. Cambridge University Press.

Fowler, H.J., Blenkinsop, S. and Tebaldi, C., 2007. Linking climate change modelling to impacts studies: recent advances in downscaling techniques for hydrological modelling. International Journal of Climatology: A Journal of the Royal Meteorological Society, 27(12), pp.1547-1578

Freeman, W.T., Jones, T.R. and Pasztor, E.C., 2002. Example-based super-resolution. IEEE Computer graphics and Applications, (2), pp.56-65.

Friederichs, P. and Hense, A., 2007. Statistical downscaling of extreme precipitation events using censored quantile regression. *Monthly weather review*, *135*(6), pp.2365-2378.

Giorgi, F. and Francisco, R., 2001. Uncertainties in the prediction of regional climate change. In *Global change and protected areas* (pp. 127-139). Springer, Dordrecht.

Giorgi, F. and Mearns, L.O., 1991. Approaches to regional climate change simulation: A review. *Rev. Geophys*, *29*(19), pp.1-2.

Gmao.gsfc.nasa.gov., 2019. MERRA-2. [online] Available at: https://gmao.gsfc.nasa.gov/reanalysis/MERRA-2/ [Accessed 9 May 2019].

Guyonnet, A., Dagoret-Campagne, S. and Mondrik, N., 2019. Local monitoring of atmospheric transparency from the NASA MERRA-2 global assimilation system. *Journal of Astronomical Instrumentation 2019*, arXiv preprint arXiv:1906.01967.

Hashmi, M.Z., Shamseldin, A.Y. and Melville, B.W., 2009, July. Downscaling of future rainfall extreme events: a weather generator based approach. In 18th World IMACS/MODSIM Congress, Cairns, Australia (pp. 13-17).

Haylock, M.R., Hofstra, N., Klein Tank, A.M.G., Klok, E.J., Jones, P.D. and New, M., 2008. A European daily high-resolution gridded data set of surface temperature and precipitation for 1950–2006. Journal of Geophysical Research: Atmospheres, 113(D20).

He, X., Chaney, N.W., Schleiss, M. and Sheffield, J., 2016. Spatial downscaling of precipitation using adaptable random forests. *Water resources research*, *52*(10), pp.8217-8237.

Hewitson, B.C. and Crane, R.G., 1996. Climate downscaling: techniques and application. Climate Research, 7(2), pp.85-95.

Hewitson, B.C. and Crane, R.G., 2006. Consensus between GCM climate change projections with empirical downscaling: precipitation downscaling over South Africa. *International Journal of Climatology: A Journal of the Royal Meteorological Society*, *26*(10), pp.1315-1337.

Hidalgo, H.G., Dettinger, M.D. and Cayan, D.R., 2008. Downscaling with constructed analogues: Daily precipitation and temperature fields over the United States. *California Energy Commission PIER Final Project Report CEC-500-2007-123*.

Hu, F., Xu, M., Yang, J., Liang, Y., Cui, K., Little, M., Lynnes, C., Duffy, D. and Yang, C., 2018. Evaluating the open source data containers for handling big geospatial raster data. *ISPRS International Journal of Geo-Information*, *7*(4), p.144.

Hu, Y., Maskey, S. and Uhlenbrook, S., 2013. Downscaling daily precipitation over the Yellow River source region in China: a comparison of three statistical downscaling methods. *Theoretical and applied climatology*, *112*(3-4), pp.447-460.

Huffman, G.J., Bolvin, D.T., Braithwaite, D., Hsu, K., Joyce, R., Xie, P. and Yoo, S.H., 2015. NASA global precipitation measurement (GPM) integrated multi-satellite retrievals for GPM (IMERG). Algorithm theoretical basis document, version, 4, p.30.

Hughes, J.P. and Guttorp, P., 1994. A class of stochastic models for relating synoptic atmospheric patterns to regional hydrologic phenomena. *Water resources research*, *30*(5), pp.1535-1546.

Idreos, S., Groffen, F., Nes, N., Manegold, S., Mullender, S. and Kersten, M., 2012. MonetDB: Two Decades of Research in Column-oriented Database Architectures. *A Quarterly Bulletin of the IEEE Computer Society Technical Committee on Database Engineering*, [online] 35(1), pp.40-45. Available at: http://sites.computer.org/debull/A12mar/monetdb.pdf [Accessed 17 Jul. 2019].

Itakura, F., 1975. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *23*(1), pp.67-72.

Jarosch, A.H., Anslow, F.S. and Clarke, G.K., 2012. High-resolution precipitation and temperature downscaling for glacier models. *Climate Dynamics*, *38*(1-2), pp.391-409.

Karl, T.R. and Trenberth, K.E., 2003. Modern global climate change. *science*, *302*(5651), pp.1719-1723.

Keogh, E. and Ratanamahatana, C.A., 2005. Exact indexing of dynamic time warping. *Knowledge and information systems*, *7*(3), pp.358-386.

Kersten, M., Zhang, Y., Ivanova, M. and Nes, N., 2011, March. SciQL, a query language for science applications. In Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases (pp. 1-12). ACM.

Kim, J., Kwon Lee, J. and Mu Lee, K., 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1646-1654).

Kim, J.W., Chang, J.T., Baker, N.L., Wilks, D.S. and Gates, W.L., 1984. The statistical problem of climate inversion: Determination of the relationship between local and large-scale climate. *Monthly weather review*, *112*(10), pp.2069-2077.

Klein, W.H. and Glahn, H.R., 1974. Forecasting local weather by means of model output statistics. *Bulletin of the American Meteorological Society*, *55*(10), pp.1217-1227.

Klein, W.H., Lewis, B.M. and Enger, I., 1959. Objective prediction of five-day mean temperatures during winter. *Journal of Meteorology*, *16*(6), pp.672-682.

Kundzewicz, Z. W., Mata, L. J., Arnell, N. W., Doll, P., Kabat, P., Jimenez, B., Miller, K., Oki, T., Zekai, S. and Shiklomanov, I. (2007) Freshwater resources and their management. In: Parry, M. L., Canziani, O. F., Palutikof, J. P., van der Linden, P. J. and Hanson, C. E. (eds.) Climate Change 2007: Impacts, Adaptation and Vulnerability. Contribution of Working Group II to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, pp. 173-210. ISBN 9780521880091

Lakshman, A. and Malik, P., 2010. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, *44*(2), pp.35-40.

Landman, W.A., Mason, S.J., Tyson, P.D. and Tennant, W.J., 2001. Statistical downscaling of GCM simulations to streamflow. Journal of Hydrology, 252(1-4), pp.221-236.

Ledesma, J.L. and Futter, M.N., 2017. Gridded climate data products are an alternative to instrumental measurements as inputs to rainfall–runoff models. Hydrological Processes, 31(18), pp.3283-3293.

Lenderink, G. and Van Meijgaard, E., 2008. Increase in hourly precipitation extremes beyond expectations from temperature changes. Nature Geoscience, 1(8), p.511.

Li, Z., Hu, F., Schnase, J.L., Duffy, D.Q., Lee, T., Bowen, M.K. and Yang, C., 2017. A spatiotemporal indexing approach for efficient processing of big array-based climate data with MapReduce. International Journal of Geographical Information Science, 31(1), pp.17-35.

Libkin, L., Machlin, R. and Wong, L., 1996, June. A query language for multidimensional arrays: design, implementation, and optimization techniques. In ACM SIGMOD Record (Vol. 25, No. 2, pp. 228-239). ACM.

Lim, K.T., Maier, S.D., Ratzesberger, O. and Zdonik, S., 2009. Requirements for science data bases and scidb. In *Conference on Innovative Data Systems Research*. [online] Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.145.1567&rep=rep1&type=pdf [Accessed 8 May 2019]

Liu, F., Lee, K., Roy, I., Talwar, V., Chen, S., Chang, J. and Ranganathan, P., 2014. GPU accelerated array queries: The good, the bad, and the promising. *HP Laboratories, Palo Alto, CA, USA, Tech. Rep. HPL-2014-50*. [online] Available at: https://liberty.princeton.edu/Publications/hpl14_gaaq.pdf [Accessed 8 May 2019]

Madden, S., 2012. From databases to big data. IEEE Internet Computing, 16(3), pp.4-6.

Marathe, A.P. and Salem, K., 1997. A language for manipulating arrays. dim, 1501, p.1.

Maraun, D., Wetterhall, F., Ireson, A.M., Chandler, R.E., Kendon, E.J., Widmann, M., Brienen, S., Rust, H.W., Sauter, T., Themeßl, M. and Venema, V.K.C., 2010. Precipitation downscaling under climate change: Recent developments to bridge the gap between dynamical models and the end user. *Reviews of Geophysics*, *48*(3).

Mehrotra, R. and Sharma, A., 2006. A nonparametric stochastic downscaling framework for daily rainfall at multiple locations. *Journal of Geophysical Research: Atmospheres*, *111*(D15).

Molod, A., Takacs, L., Suarez, M. and Bacmeister, J., 2015. Development of the GEOS-5 atmospheric general circulation model: Evolution from MERRA to MERRA2. Geoscientific Model Development, 8(5), pp.1339-1356.

Monetdb.org., n.d. Storage model | MonetDB. [online] Available at: https://www.monetdb.org/Documentation/Manuals/MonetDB/Architecture/Storagemodel [Accessed 6 Aug. 2019].

Najafi, M.R., Moradkhani, H. and Wherry, S.A., 2010. Statistical downscaling of precipitation using machine learning with optimal predictor selection. *Journal of Hydrologic Engineering*, *16*(8), pp.650-664.

NASA EARTHDATA, 2019. *Earthdata Cloud Evolution | Earthdata*. [online] Available at: https://Earthdata.nasa.gov/eosdis/cloud-evolution [Accessed 17 Jul. 2019].

Nashwan, M.S., Shahid, S. and Chung, E.S., 2019. Development of high-resolution daily gridded temperature datasets for the central north region of Egypt. *Scientific data*, 6(1), p.138.

Naydenova, I. and Kaloyanova, K., 2010. Sparsity Handling and Data Explosion in OLAP Systems. *MCIS*, *10*, pp.62-70.

Noor, M., bin Ismail, T., Ullah, S., Iqbal, Z., Nawaz, N. and Ahmed, K., 2019. A non-local model output statistics approach for the downscaling of CMIP5 GCMs for the projection of rainfall in Peninsular Malaysia. *Journal of Water and Climate Change*. [online] Available at: https://doi.org/10.2166/wcc.2019.041 [Accessed 12 Nov. 2019].

Obe, R. and Hsu, L., 2011. PostGIS 2.0 3D and Raster support enhancements. North Carolina GIS Conference. [online] Available at: https://www.postgis.us/downloads/ncgis2011/NCGISSDBPostGIS20_2011.pdf [Accessed 1 Nov. 2019].

Peleg, N., Molnar, P., Burlando, P. and Fatichi, S., 2019. Exploring stochastic climate uncertainty in space and time using a gridded hourly weather generator. Journal of Hydrology, 571, pp.627-641.

Planthaber, G., Stonebraker, M. and Frew, J., 2012, November. EarthDB: scalable analysis of MODIS data using SciDB. In Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data(pp. 11-19). ACM.

Posadas, A., Duffaut Espinosa, L.A., Yarlequé, C., Carbajal, M., Heidinger, H., Carvalho, L., Jones, C. and Quiroz, R., 2015. Spatial random downscaling of rainfall signals in Andean heterogeneous terrain. *Nonlinear Processes in Geophysics*, *22*(4), pp.383-402.

Pour, S.H., Shahid, S., Chung, E.S. and Wang, X.J., 2018. Model output statistics downscaling using support vector machine for the projection of spatial and temporal changes in rainfall of Bangladesh. *Atmospheric research*, *213*, pp.149-162.

Qi, Z. and Xuelong, L., 2019. Big data: new methods and ideas in geological scientific research. Big Earth Data, 3(1), pp.1-7.

Rau, M., He, Y., Goodess, C. and Bárdossy, A., 2019. Statistical downscaling to project extreme hourly precipitation over the United Kingdom. International Journal of Climatology. [online] Available at: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/joc.6302 [Accessed 17 Aug. 2019]

Rebora, N., Ferraris, L., von Hardenberg, J. and Provenzale, A., 2006. RainFARM: Rainfall downscaling by a filtered autoregressive model. *Journal of Hydrometeorology*, *7*(4), pp.724-738.

Reiner, B., Hahn, K., Höfling, G. and Baumann, P., 2002, September. Hierarchical storage support and management for large-scale multidimensional array database management systems. In International Conference on Database and Expert Systems Applications (pp. 689-700). Springer, Berlin, Heidelberg.

Richardson, C.W., 1981. Stochastic simulation of daily precipitation, temperature, and solar radiation. *Water resources research*, *17*(1), pp.182-190.

Salathé Jr, E.P., 2003. Comparison of various precipitation downscaling methods for the simulation of streamflow in a rainshadow river basin. *International Journal of Climatology: A Journal of the Royal Meteorological Society*, *23*(8), pp.887-901.

Saunders, K., Stephenson, A.G., Taylor, P.G. and Karoly, D., 2017. The spatial distribution of rainfall extremes and the influence of El Niño Southern Oscillation. *Weather and climate extremes*, *18*, pp.17-28.

Scied.ucar.edu., 2011. Climate Modeling | UCAR Center for Science Education. [online] Available at: https://scied.ucar.edu/longcontent/climate-modeling [Accessed 3 Sep. 2019].

Sheehan, M.C., Fox, M.A., Kaye, C. and Resnick, B., 2017. Integrating health into local climate response: Lessons from the US CDC Climate-Ready States And cities initiative. Environmental health perspectives, 125(9), p.094501.

Sobhani, M., Campbell, A., Sangamwar, S., Li, C. and Hong, T., 2019. Combining Weather Stations for Electric Load Forecasting. *Energies*, *12*(8), p.1510.

Soleh, A.M., Wigena, A.H., Djuraidah, A. and Saefuddin, A., 2015. Statistical downscaling to predict monthly rainfall using linear regression with L1 regularization (LASSO). *Applied Mathematical Sciences*, *9*(108), pp.5361-5369.

Song, J., Guo, C., Wang, Z., Zhang, Y., Yu, G. and Pierson, J.M., 2015. HaoLap: a Hadoop based OLAP system for big data. Journal of Systems and Software, 102, pp.167-181.

Soroush, E. and Balazinska, M., 2013, April. Time travel in a scientific array database. In Data Engineering (ICDE), 2013 IEEE 29th International Conference on (pp. 98-109). IEEE.

Sørup, H.J.D., Christensen, O.B., Arnbjerg-Nielsen, K. and Mikkelsen, P.S., 2016. Downscaling future precipitation extremes to urban hydrology scales using a spatio-temporal Neyman–Scott weather generator. Hydrology and Earth System Sciences, 20, pp.1387-1403.

Stonebraker, M., Brown, P., Zhang, D. and Becla, J., 2013. SciDB: A database management system for applications with complex analytics. Computing in Science & Engineering, 15(3), pp.54-62.

Stonebraker, M., Duggan, J., Battle, L. and Papaemmanouil, O., 2013. SciDB DBMS research at MIT. IEEE Data Eng. Bull., 36(4), pp.21-30.

Tan, C.W., Webb, G.I. and Petitjean, F., 2017, June. Indexing and classifying gigabytes of time series under time warping. In *Proceedings of the 2017 SIAM international conference on data mining* (pp. 282-290). Society for Industrial and Applied Mathematics.

Tan, Z. and Yue, P., 2016, July. A comparative analysis to the array database technology and its use in flexible VCI derivation. In Agro-Geoinformatics (Agro-Geoinformatics), 2016 Fifth International Conference on (pp. 1-5). IEEE.

Tatli, H., Nüzhet Dalfes, H. and Sibel Menteş, Ş., 2004. A statistical downscaling method for monthly total precipitation over Turkey. International Journal of Climatology: A Journal of the Royal Meteorological Society, 24(2), pp.161-180.

Terzago, S., Palazzi, E. and Hardenberg, J.V., 2018. Stochastic downscaling of precipitation in complex orography: a simple method to reproduce a realistic fine-scale climatology. *Natural Hazards and Earth System Sciences*, *18*(11), pp.2825-2840.

Turco, M., Llasat, M.C., Herrera, S. and Gutiérrez, J.M., 2017. Bias correction and downscaling of future RCM precipitation projections using a MOS-Analog technique. Journal of Geophysical Research: Atmospheres, 122(5), pp.2631-2648.

Turco, M., Quintana-Seguí, P., Llasat, M.C., Herrera, S. and Gutiérrez, J.M., 2011. Testing MOS precipitation downscaling for ENSEMBLES regional climate models over Spain. Journal of Geophysical Research: Atmospheres, 116(D18).

van Ballegooij, A., Cornacchia, R., de Vries, A.P. and Kersten, M., 2005, August. Distribution rules for array database queries. In International Conference on Database and Expert Systems Applications (pp. 55-64). Springer, Berlin, Heidelberg.

van Ballegooij, A.R., 2004, March. RAM: a multidimensional array DBMS. In International Conference on Extending Database Technology (pp. 154-165). Springer, Berlin, Heidelberg.

Vandal, T., Kodra, E., Ganguly, S., Michaelis, A., Nemani, R. and Ganguly, A.R., 2017, August. Deepsd: Generating high resolution climate change projections through single image super-resolution. In *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1663-1672). ACM.

Vandal, T.J., 2018. Statistical Downscaling of Global Climate Models with Image Super-resolution and Uncertainty Quantification. Ph.D. Northeastern University

Verdin, A., Rajagopalan, B., Kleiber, W., Podestá, G. and Bert, F., 2018. A conditional stochastic weather generator for seasonal to multi-decadal simulations. Journal of Hydrology, 556, pp.835-846.

Von Storch, H., Zorita, E. and Cubasch, U., 1993. Downscaling of global climate change estimates to regional scales: an application to Iberian rainfall in wintertime. *Journal of Climate*, *6*(6), pp.1161-1171.

Vrac, M. and Naveau, P., 2007. Stochastic downscaling of precipitation: From dry events to heavy rainfalls. *Water resources research*, *43*(7).

Vrac, M., Stein, M. and Hayhoe, K., 2007. Statistical downscaling of precipitation through nonhomogeneous stochastic weather typing. Climate Research, 34(3), pp.169-184.

Walton, D.B., Sun, F., Hall, A. and Capps, S., 2015. A hybrid dynamical–statistical downscaling technique. Part I: Development and validation of the technique. *Journal of Climate*, *28*(12), pp.4597-4617.

Wang, J. and Zhang, X., 2008. Downscaling and projection of winter extreme daily precipitation over North America. *Journal of Climate*, *21*(5), pp.923-937.

Wetterhall, F., Bárdossy, A., Chen, D., Halldin, S. and Xu, C.Y., 2006. Daily precipitation-downscaling techniques in three Chinese regions. *Water Resources Research*, *42*(11).

Wetterhall, F., Halldin, S. and Xu, C.Y., 2005. Statistical precipitation downscaling in central Sweden with the analogue method. *Journal of Hydrology*, *306*(1-4), pp.174-190.

Wetterhall, F., Pappenberger, F., He, Y., Freer, J. and Cloke, H.L., 2012. Conditioning model output statistics of regional climate model precipitation on circulation patterns. Nonlinear Processes in Geophysics, 19(6), pp.623-633.

Wigley, T.M.L., Jones, P.D., Briffa, K.R. and Smith, G., 1990. Obtaining sub-grid-scale information from coarse-resolution general circulation model output. *Journal of Geophysical Research: Atmospheres*, *95*(D2), pp.1943-1953.

Wilby, R.L. and Wigley, T.M.L., 1997. Downscaling general circulation model output: a review of methods and limitations. *Progress in physical geography*, *21*(4), pp.530-548.

Wilby, R.L., 1998. Statistical downscaling of daily precipitation using daily airflow and seasonal teleconnection indices. *Climate Research*, *10*(3), pp.163-178.

Wilby, R.L., Charles, S.P., Zorita, E., Timbal, B., Whetton, P. and Mearns, L.O., 2004. Guidelines for use of climate scenarios developed from statistical downscaling methods. *Supporting material of the Intergovernmental Panel on Climate Change, available from the DDC of IPCC TGCIA*, *27*. [online] Available at: http://apps.ipcc-data.org/guidelines/dgm_no2_v1_09_2004.pdf [Accessed 5 May 2019]

Wilby, R.L., Conway, D. and Jones, P.D., 2002. Prospects for downscaling seasonal precipitation variability using conditioned weather generator parameters. Hydrological Processes, 16(6), pp.1215-1234.

Wilby, R.L., Wigley, T.M.L., Conway, D., Jones, P.D., Hewitson, B.C., Main, J. and Wilks, D.S., 1998. Statistical downscaling of general circulation model output: A comparison of methods. *Water resources research*, *34*(11), pp.2995-3008.

Wilks, D.S., 1995. *Statistical Methods in the Atmospheric Sciences: An Introduction*. Academic press.

Wilks, D.S., 2010. Use of stochastic weathergenerators for precipitation downscaling. *Wiley Interdisciplinary Reviews: Climate Change*, *1*(6), pp.898-907.

Willems, P. and Vrac, M., 2011. Statistical precipitation downscaling for small-scale hydrological impact investigations of climate change. *Journal of hydrology*, *402*(3-4), pp.193-205.

Wong, G., Maraun, D., Vrac, M., Widmann, M., Eden, J.M. and Kent, T., 2014. Stochastic model output statistics for bias correcting and downscaling precipitation including extremes. Journal of Climate, 27(18), pp.6940-6959.

Xu, C.Y., 1999. Climate change and hydrologic models: A review of existing gaps and recent research developments. *Water Resources Management*, *13*(5), pp.369-382.

Yang, C., Huang, Q., Li, Z., Liu, K. and Hu, F., 2017. Big Data and cloud computing: innovation opportunities and challenges. *International Journal of Digital Earth*, 10(1), pp.13-53.

Yang, C., Yu, M., Li, Y., Hu, F., Jiang, Y., Liu, Q., Sha, D., Xu, M. and Gu, J., 2019. Big Earth data analytics: A survey. *Big Earth Data*, 3(2), pp.83-107.

Yu, M., Bambacus, M., Cervone, G., Clarke, K., Duffy, D., Huang, Q., Li, J., Li, W., Li, Z., Liu, Q. and Resch, B., 2020. Spatiotemporal event detection: a review. *International Journal of Digital Earth*, pp.1-27.

Zhang, Y., Kersten, M., Ivanova, M. and Nes, N., 2011, September. SciQL: bridging the gap between science and relational DBMS. In *Proceedings of the 15th Symposium on International Database Engineering & Applications* (pp. 124-133). ACM.

Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B. and Fu, Y., 2018. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 286-301).

BIOGRAPHY

Mengchao Xu graduated from the High School Affiliated to Nanjing Normal University, Nanjing, China, in 2009. He received his Bachelor of Arts from Saint John's University (MN) in 2013 and completed his Master of Sciences study in Geographic Information Science at University of Redlands in 2014.