CRANFIELD UNIVERSITY

Wenting Deng

A Grid-based GIS System for Spatial Analysis

SCHOOL OF ENGINEERING

MSc THESIS

CRANFIELD UNIVERSITY
School of Engineering

MSc THESIS
ACADEMIC YEAR 2009-2010

Wenting Deng

A Grid-based GIS System for Spatial Analysis

Supervisor: Dr. Sining Wu

August 2010

This thesis is submitted in partial fulfillment of the requirements
for the degree of Master of Science

Acknowledgements

First and foremost, I would like to express my respect to my supervisor Dr Sining Wu for his guidance throughout the whole project. Thanks for my supervisor Prof. Jiansong Li for the knowledge he shared with me.

I also want to give thanks to Kath Tipping and Peter Sherar and the rest of the lecturers from Applied Mathematics and Computing Group department for the hard working during the course. I really learn a lot from you.

I am grateful to friends and colleagues over the year with whom I have studied with for the time we took together.

And above all, to my parents for their love and faith

Abstract

This work presents a GIS system for the spatial analysis using grid computing technologies. Spatial analysis is an important function of GIS system. The purpose of this project is to prove that the grid computing technology and parallel computing technology can be used for spatial analysis.

As the spatial analysis is a complex conception with a series of different categories. Only one method is implemented in this project, which is a measurement of spatial autocorrelation – the Moran's I.

The whole project consists of four main parts: 1. Test datasets generation; 2. Serial and parallel program implementation; 3. Tests in Astral environment; 4. Tests in Grid environment.

In part 1, a single program is implemented for data generation. It is used to generate raster datasets with different grid sizes, spatial patterns, and attribute values.

In part 2, both serial program and parallel program are written with C++. For both of the programs, the source datasets and the results are stored in .txt file, and the results are represented with Gnuplot. The results will be checked with ESRI ArcGIS and GeoDa software.

In part 3, the parallel program written with MPI will be tested in Astral environment. Different datasets will be used, and we get a conclusion that the performance is improved significantly by using with MPI.

In part 4, a virtual Grid environment is set up by using Globus middleware and Sun VirtualBox. In this environment, a client asks for running spatial analysis jobs, and the jobs are submitted in the Grid environment and run on a remote or local machine, the results will be returned to the client after the calculation.

In summary, spatial analysis benefits from grid computing technologies and parallel computing technologies.

Contents

List of Figures

List of Tables

1. Introduction

GIS (Geographic Information System) is a system used to capture, store, analyze, manage, and present data which contains coordinate information. Spatial analysis is the process of applying analytical techniques to the geographically referenced datasets so that new information is generated. It is one of the most important functions of GIS system. The first application of spatial analysis is considered as the John Snow's map. John Snow is a doctor who depicted a cholera outbreak in London in 1954 and found out the source of the disease, which is a public water pump on Broad Street. Figure 1 shows the John Snow's map.

Nowadays, spatial analysis is increasingly being applied to solve large, realistic problems. The analysis often requires the use of statistical methods that are always computationally intensive. In this project, we want to use corresponding technologies to improve the performance of spatial analysis. Spatial analysis contains different types, such as: spatial autocorrelation; spatial interpolation; spatial regression; spatial interaction; simulation and modeling. In this project we use a case study to prove our thoughts: the Moran's I – a kind of the measurements of spatial autocorrelation.

Moran's I is a measurement of spatial autocorrelation. For a dataset with n grid cells, any two grid cells should be calculated for at least once to get the results. In the process of calculations, an n×n spatial weight matrix is generated to store the spatial relationship between any two grid cells in the dataset. Consequently, this analysis is memory intensive. As the calculation must be performed for each grid cell, the analysis is also computing intensive. How to use existing resource to finish the spatial analysis process is becoming a problem, especially when it is applied to large datasets. There are three useful solutions; all of them will be tested in this project.

- Efficient loop structures
- Parallel computing
- Using Grid resource

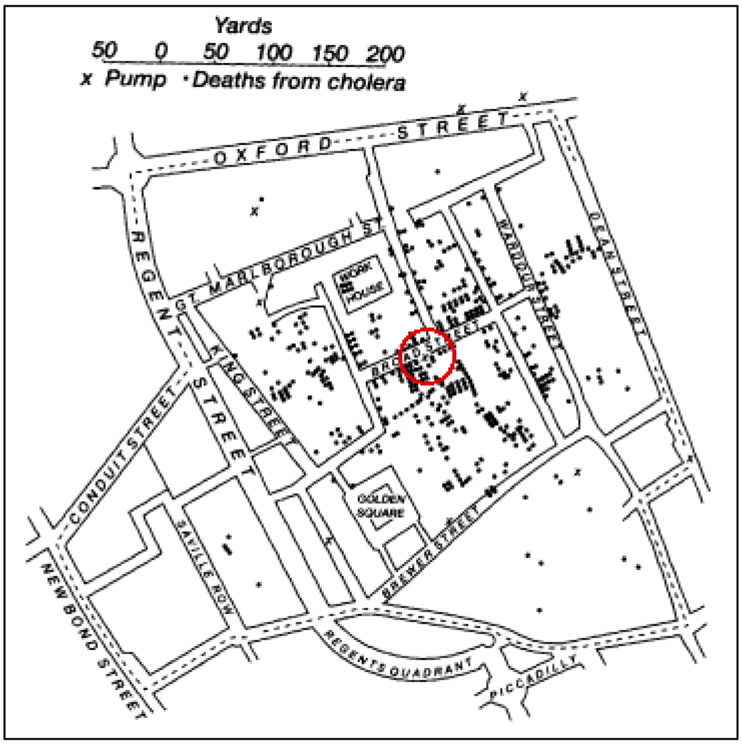


Figure 1 John Snow's map

1.1 Thesis overview

The aim of this project is to prove that spatial analysis benefits from grid computing and parallel computing. A case study is implemented in this project, which is the calculation of the Moran's I, a measurement of spatial autocorrelation. In order to achieve this goal, we need to implement a serial program first of all, and then a parallel program will be built. A series of tests should be done in Astral environment and Grid environment.

Tests in Astral environment is aim to prove that the parallel program which is written with MPI improves the calculation performance significantly. A domain decomposition method is taken which divides the spatial weight matrix into strips. As the strips are independent on each other, there is no communications in the process of calculations. The only communications arise at the beginning and the end of the whole process. At the beginning of the process, the program needs to broadcast dataset to all of the processors, and at the end of the process, the program needs to gather the results of each processor together. As a result, the speed-up ratios are near to the numbers of processors.

In order to do tests in grid environment, we should first build up a grid environment. In this project, we use Globus middleware and Sun VirtualBox to build up three different virtual machines to simulate the grid environment. In this grid environment there are two server machines and one client machine. One of the server machines needs to install Globus Toolkits and Simple CA. This is the manager of the certificates in the environment. The other server machine needs to install Globus Toolkit. On the server machines, GridFTP, RFT (Reliable File Transfer), and GRAM (Grid Resource Allocation and Management) services must be setup. The third machine acts as a client who sends the request of submitting the job in the grid environment and gathers results.

The results are stored in a .txt file. And the results are plotted with Gnuplot. This will give the users a more clear view of what the results look like.

1.2 Thesis outline

The thesis report is organized as follows.

In Chapter Moran Spatial Statistic the details of Moran spatial statistic (Moran’s I) are introduced. We also introduce the fundamental of hypothesis tests.

In Chapter Algorithm Implementation the implementations of both serial algorithm and parallel algorithm will be introduced. The implementation of data generation program will be first introduced. And finally visualize the results with Gnuplot.

In Chapter Tests in Astral Environment the details about the tests in Astral environment will be introduced.

In Chapter Tests in Grid Environment a virtual gird environment will be set up by Globus middleware and Sun VirtualBox first of all. And the tests will be done in this simulated grid environment.

For both of the tests chapters, a conclusion will be given at the end of each chapter, which will analyze the system performance.

The last chapter consists of the final conclusions and further work.

2. Literature Review

The main aim of this literature review is to review the basic conceptions in this project, and introduce appropriate methods used to improve the performance of spatial analysis. From the Introduction section, we know that most of the spatial analysis methods are considered as memory intensive applications as well as computing intensive applications. And, parallel computing technologies and grid computing technologies will always be accepted as the solution for these applications. Consequently, grid computing and parallel computing as well as the usages in GIS system will be introduced next.

2.1 Grid Computing

Grid computing has been regarded as the next generation of World Wide Web (WWW), which provides an infrastructure of sharing computing resources distributed across the whole world. The ideas of the Grid were brought together by Ian Foster, Carl Kesselman and Steve Tuecke, who are widely regarded as the "fathers of the Grid". They also led to the effort to create the Globus Toolkit, an open source software toolkit used for building Grid systems and applications, which is being mainly developed by the Globus Alliance.

The earliest definition of Grid is considered as the definition from Ian Foster et al, who defined as "A computational grid is a hardware and software infrastructure that provides dependable consistent pervasive and inexpensive access to high-end computational capabilities."[1] From this definition, we see that they suggested a form of on-demand access to computational resources. And this idea was first introduced by Len Kleinrock in 1969: "We will probably see the spread of 'computer utilities', which like present electric and telephone utilities, will service individual homes and offices across the country."

Then Ian Foster gave the definition of Grid from another perspective. He provided the definition of "Grid problem". It is defined as "flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources – what we refer to as virtual organizations."[2] And he suggested that "The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations."[2] In the same article, he and his colleagues also provide a description of the Grid architecture with five layers: Grid Fabric layer; Connectivity layer; Resource layer; Collective layer and Application layer.

In order to specify the details of Grid, Ian Foster also suggested a simple checklist of a Grid system in a report "What is the Grid? A Three Point Checklist" written in 2002:

- Coordinates resources that are not subject to centralized control.
- Using standard, open, general-purpose protocols and interfaces.
- To deliver nontrivial qualities of service.

From the checklist shown above, we know that some systems cannot be considered as a Grid, such as: Sun Grid Engine (it has a centralized control). The first real Grid system is considered as I-WAY (Information Wide-Area Year) a national grid built for the project SC95 (Super Computing 95).[3] There are also some examples of Grids: UK e-Science Grid; TeraGrid; and EC Data Grid. Ongoing developments with Grid computing are addressing the solution of dynamic problem with high-throughput which is based on "virtual organizations".

In summary, Grid computing principles focus on large-scale resource sharing in distributed systems in a flexible, secure, and coordinated fashion.[4]

2.2 Parallel Computing

Parallel computing is the Computer Science discipline that deals with the system architecture and software issues with a form of computation in which the calculations are carried out simultaneously.

The interest of parallel computing dates back to the late 1950s. In the 1954, IBM introduced 704, the first commercial machine with floating-point hardware with Gene Amdahl as one of the principal architects. Throughout the 1960s and 1970s, advanced improvements have been done in the area of supercomputers. Although, the performance has magnitude increase every five years, but there are minor adaptations of the Von Neumann model. Here we must talk about Flynn, who developed taxonomy that classifies parallel computer architectures into two categories: single instruction multiple data (SIMD) stream and multiple instruction multiple data (MIMD) stream.[5] Starting in 1980s, clusters have been applied in many areas, and they are widely used nowadays. The ranking of the top 500 powerful computer systems illustrates that the cluster systems dominate the supercomputer area today. A more detailed introduction of the history of parallel computing is found on a website built by Virginia Tech/Norfolk State University.

The primary reasons for using parallel computing are:

- To save time for the execution of programs so that results are obtained in a reasonable time.
- To overcome memory constraints for large problems.
- To provide concurrency which allows to do multiple things at the same time
- Use of non-local resources
- To save cost by using existing multiple computing resources

According to Vivek Sarkar three fundamental problems must be solved when developing a parallel algorithm: identification of parallelism; partitioning of the problem into a set of sequential tasks; and scheduling the tasks.[6] There are four types of parallelism: trivial parallelism; pipeline parallelism; data parallelism; and functional parallelism. The last two kinds of parallelism will be discussed in Parallel Algorithm section.

In parallel computing speed-up ratio and efficiency are two important measurements of the parallel algorithms’ performance. Generally speaking, the factors limiting the performance include: software overhead; load balancing; communication overhead; and Amdahl’s Law.

In general, the goal of parallel computing is to decrease the computing time and solve large problems by using multiple processors.

2.3 Combination with GIS

According to Dueker and Kjerene's definition, GIS is a system of hardware, software, data, people, organizations and institutional arrangements for collecting, storing, analyzing, and disseminating information about areas of the earth.[7] As the development of GIS, the data used by GIS system tend to be geographically distributed and increase in size, and the spatial analysis used in GIS system tends to be more complex and computationally intensive.

The development of GIS sciences and technologies motivate the concern of the next generation of GIS, including multi-resources distributed, high-performance computation and data transfer, and collaborative platform of virtual organization for multiple end users.[8] Some researchers consider Grid computing as the backbone of the development of GIS. This is because Grid computing technology represents a new approach to collaborative computing and problem solving in data intensive and computationally intensive environment and has the chance to satisfy all the requirements of a distributed, high-performance and collaborative GIS. First, Grid makes all the distributed computing resources into an integrated environment. This capability makes it possible to access to the distributed data used by GIS systems. Second, Grid computing provides high performance computing capability, which will satisfy the needs of the more complex and computationally intensive spatial analysis used in GIS systems. Third, Grid computing contains GSI (Grid Security Infrastructure) which will provide the security issues for a distributed GIS. GSI makes sure that the sharing and transfer of spatial data are secure.

Grid applications in geographic information analysis have lagged, mainly due to the lack of software that is used to leverage geographic information analysis applications to take advantage of Grid resources.[9] But researchers have contributed a lot in this area. Shekhar et al. have developed parallel algorithms for spatial analysis.[10] Hunsaker et al. developed a parallel implementation for spatial data error simulation.[11] Puppo summarized progress towards implementing parallel terrain modeling.[12] Shaowen Wang et al. built a Grid-based GIS system for geographic information analysis.[9][13] Qinghui Sun et al. also tried to design a middleware for Grid based GIS.[8]

As a conclusion, GIS benefits from parallel and Grid computing, and Grid computing has the chance to lead GIS into a new “Grid-enabled GIS” age in terms of computing paradigm, resource sharing pattern, and online collaboration.

2.4 Introduction of Moran’s I

Moran’s I is a measure of spatial autocorrelation developed by Patrick A. P. Moran[14] which is used to indicate the spatial clustering. It is expressed as:

$$I = \frac{n}{S_0} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\left(x_i - \overline{x}\right)\left(x_j - \overline{x}\right)}{\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2}$$

As clustering is a global property of a dataset, in order to get a local indicator of spatial autocorrelation, Anselin developed a class of local indicators of spatial association in 1995[15] which contains a local version of Moran’s I. The local Moran’s I can be used to indicate the spatial clusters and outliers. The spatial clusters and outliers are local property of a dataset. It is expressed as:

$$I_i = z_i \sum_{j=1}^{n} w_{ij} z_j \qquad z_i = \frac{\left(x_i - \overline{x}\right)}{\delta}$$

Both of global Moran’s I and local Moran’s I are constructed by combining an indicator of attribute similarity and an indicator of spatial similarity.[16] The attribute similarity is expressed as a cross product between attribute value of object i and object j (x_i and x_j). The spatial similarity is expressed as a spatial weight matrix with element w_{ij} that stands for the relationship between object i and object j.

The appropriate choice of the spatial weight matrix is one of the most difficult and controversial methodological issues in exploratory spatial data analysis and spatial econometrics.[17] Getis suggested that three ways of thinking about the type of spatial weights matrix existed: theoretical; topological; and empirical.[18] Moran is the first one who introduced the term of “spatial weights matrix” around 1950.[14] He provided “Contiguity Spatial Weight Matrix” and “Distance-based Spatial Weight Matrix”. In 1968, Berry and Marble published a book, in which they suggested “Queen’s Spatial Weight Matrix” and “K-Nearest Spatial Weight Matrix”.[19] Cliff and Ord provided a spatial weight matrix by using a combination of distance measure and relative length of the boundary between objects. Similarly, Dacey introduced another spatial weight matrix named behind him: “Dacey Spatial Weight Matrix” in 1968 which takes the relative area of the objects into account. [19] These spatial weight matrices will be introduced and compared lately in Chapter Moran Spatial Statistic.

2.5 Summary

In this section after the brief overview of Grid computing and parallel computing we focused on the combination of Grid and GIS and the introduction of Moran’s I. We think that GIS benefits from parallel and Grid computing, and Grid computing has the chance to lead GIS into a new “Grid-enabled GIS” age. By using of Grid computing and parallel computing, spatial analysis benefits a lot, such as accessing to distributed data and improving the performance of calculation. Finally we gave a brief introduction of Moran’s I which is used as a case study in this thesis.

From this section, we get a conclusion that parallel computing and Grid computing can be used for GIS, especially for GIS spatial analysis. And this conclusion will instruct us for this thesis project.

3. Moran Spatial Statistic

The Moran spatial statistic which is also referred as the Moran's I or the Moran's Index. The Moran's I consists of two different versions: global version and local version. A global version which is the Global Moran's I and a local version which is the Local Moran's I. In this chapter, it gives an introduction of the basic ideas of both the Global Moran's I and Local Moran's I, contains the formula of calculating the Global and Local Moran's I statistic values, formulas of calculating the z-score and p-value of the Global and Local Moran's I, constructing the spatial weight matrix which is needed during the computations.

3.1 Introduction

In statistics, Moran's I is a measure of spatial autocorrelation developed by Patrick A. P. Moran. [14] The term developed by Moran stands for Global Moran's I which indicates the spatial clustering. Clustering is a global property of the spatial pattern in a dataset (values and/or locations are more closely together than they would be randomly).[16] In order to improve the spatial autocorrelation theory, Anselin developed a class of local indicators of spatial association in 1995 which is used to detect the local spatial clusters and outliers. Local Moran's I is one of the local indicators. Local spatial clusters, sometimes referred to as hot spots, may be identified as those locations or sets of contiguous locations for which the LISA is significant.[15] And local spatial outliers identify the locations with high values surrounded with the low values or the locations with low values surrounded with the high values.

3.2 Fundamental of Hypothesis Tests

The process of computing the Moran's I statistic contains the hypothesis tests. The hypothesis is an assumption about a phenomena, this assumption can be true or false. There are two kinds of the hypotheses, one is null hypothesis, and other is alternative hypothesis. The precise specification of the null and alternative hypotheses will clearly depend on the variables under investigation. [20]

The null hypothesis expressed as H_0, indicates that the observations are random. In the Moran's I case, it stands for an assumption of the spatial randomness. The null hypothesis is always assumed to be correct until or unless shown to be unacceptable as a result of the statistical test. [20]

The alternative hypothesis expressed as H_α, indicates that the observations may be effected by a certain factor. In the Moran's I case, it stands for an assumption of spatial autocorrelation. α stands for the level of significance, it is called significant level. The decisions are made under a specific significant level. For example, if the observed statistics is located beyond [-1.96, 1.96], we may conclude that the observed statistics is statistically significant with a significant level of 0.05.

There are two types of errors in the hypothesis testing: Type I error is rejecting the null hypothesis H_0 when it is true. The probability of the Type I error equals to α. Type II error is accepting the null hypothesis H_0 when it is false. The probability of the Type II error equals to β. The relationship between α and β is very interesting. When α becomes smaller, β will become bigger, when α becomes bigger, β will become smaller. You cannot reduce the probability of the two types of error at the same time.

With the definition of the region of rejection, there are one-tailed test and two-tailed test. The Figure 2 below indicates the differences between the two tests. In this project, two-tailed test is chosen.

About null hypotheses, there are two basic types, the randomization null hypothesis and the normalization null hypothesis. The randomization null hypothesis is often used when the distribution type is unknown. It postulates that the observed spatial pattern of your data represents one of many (n!) possible spatial arrangements. This is the hypothesis test used for local Moran's I. The normalization null hypothesis postulates that the observed values are derived from an infinitely large. It is used for global Moran's I.

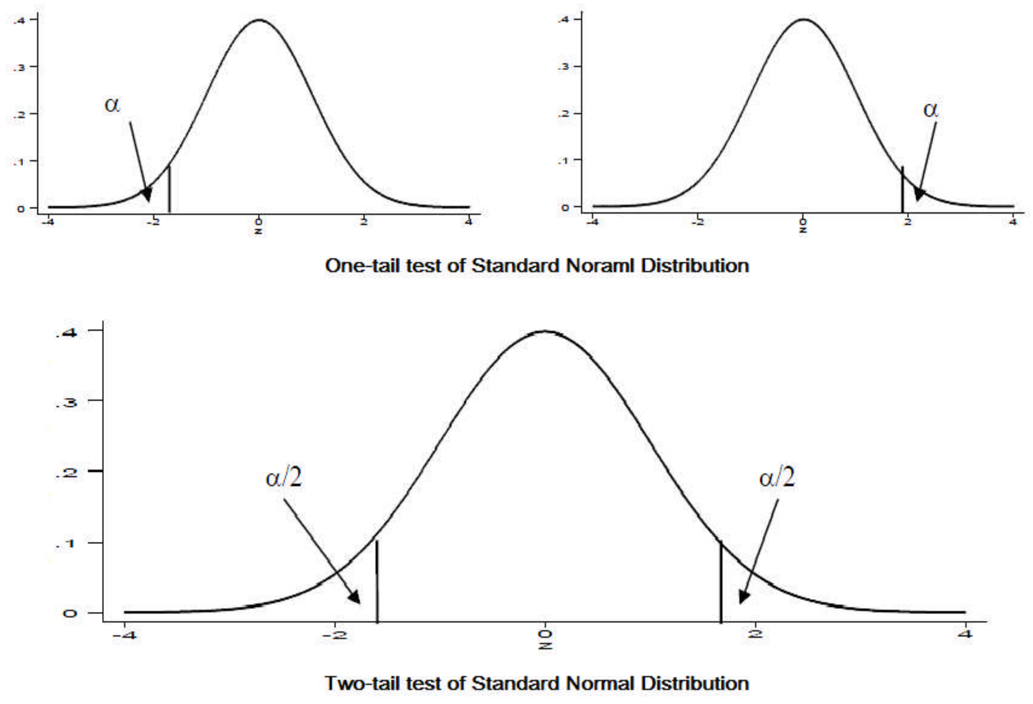


Figure 2.One-tail test and Two-tail test

3.3 Global Moran Spatial Statistic

Moran's I is the most familiar global spatial autocorrelation statistic.

Moran's I is defined as [14]

$$I = \frac{n}{S_0}\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\left(x_i - \overline{x}\right)\left(x_j - \overline{x}\right)}{\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2} \qquad S_0 = \sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} \tag{1}$$

x_i is the attribute for object i, $\overline{x}$ is the mean value of the attributes, n is the total number of the objects, S_0 is the sum of all the elements in the spatial weight matrix.

Moran's I varies between -1 and 1. It measures whether the dataset is clustered, dispersed, or random. On a given significant level, a value near 1 indicates that similar attributes are clustered, and a value near -1 indicates that dissimilar attributes are clustered. [21] If the value of Moran's I is close to 0, it indicates that the dataset has a random pattern.

There is another term which is important during the computation of the Moran's I, which is the 'spatial lag'. The spatial lag is defined as: [17]

$$Wz_i = \sum_{j=1}^{n} w_{ij} z_j \tag{2}$$

for which the value at each location is obtained as a weighted average of the values at the neighboring locations. For most of the applications, the z_i used in formula (2) does not stand for the deviation from the mean value or the raw value of the observation, but stands for the standardized variables. The using of the spatial lag facilitates the computation of the Moran's I statistic. It is also used to draw the Moran Scatter plot. In a Moran Scatter plot, the spatial lag (weighted average of values at neighboring locations) is plotted against the value at each location and the slope of the regression line through this scatter corresponds to the familiar Moran's I statistics for spatial autocorrelation.[22] In other words, a scatter plot is generated with z_i value on the horizontal axis and its spatial lag Wz_i on the vertical axis. The Moran Scatter plot is used to visualize the spatial autocorrelation, as it contains the Moran's I statistic. Although this is a very useful variable for the computation of the Moran's I, some of the commercial GIS software do not accept this variable as one of the output values. For example, in ESRI ArcGIS, only the Global Moran's I, z-score, and p-value are written into the result window.

3.4 Local Moran Spatial Statistic

In 1995, Anselin outlines a class of local indicators of spatial association (LISA), which contains a series of local indicators, such as local Gamma, local Moran statistic, and local Geary. He suggests that "A local indicator of spatial association (LISA) is any statistic that satisfies the following tow requirements: a. the LISA for each observation gives an indication of the extent of significant spatial clustering of similar values around that observation; b. the sum of LISAs for all observations is proportional to a global indicator of spatial association."

Local Moran's I is exploited by Anselin in 1995. A local version of Moran's I allows for the detection of spatial clusters and outliers.[16] It gives an indication of extent of significant spatial clustering of similar values around that observation. It is used to investigate local spatial clusters and as a diagnostic for outliers with respect to the measure of global association. In addition to significance, the local Moran also indicates the type of local spatial autocorrelation, either clusters (of high or low values) or outliers (high-low or low-high). [16]

According to Luc Anselin, a local Moran statistic for an observation value i is defined as

$$I_i = z_i \sum_{j=1}^{n} w_{ij} z_j \qquad z_i = \frac{(x_i - \overline{x})}{\delta} \tag{3}$$

where the observation z_i is the standardized scores of attribute value at i. δ is the standard deviation of the set of the attribute values. The weights w_{ij} may be in row standardized form, though this is not necessary, and by convention $w_{ii} = 0$.[15]

A positive I_i means either a high value surrounded by high values (high - high) or a low value surrounded by low values (low - low). A negative I_i means either a low value surrounded by high values (low - high) or a high value surrounded by low values (high - low).[21]

According to Anselin, the sum of LISAs should be proportional to a global indicator, so we have formula below:

$$\sum_{i=1}^{n} I_i = \gamma I \tag{4}$$

$$\sum_{i=1}^{n} I_i = \sum_{i=1}^{n} z_i \sum_{j=1}^{n} w_{ij} z_j = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} z_i z_j \tag{5}$$

According to the definition of z_i, we continue write the formula below:

$$I = \frac{n}{S_0} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} z_i z_j}{\sum_{i=1}^{n} z_i^{2}} = \frac{\sum_{i=1}^{n} I_i}{\frac{S_0}{n} \sum_{i=1}^{n} z_i^{2}} \tag{6}$$

From formula (4) and formula (6), formula (7) are derived

$$\gamma = S_0 m_2 \qquad m_2 = \frac{1}{n} \sum_{i=1}^{n} z_i^{2} \tag{7}$$

If the spatial weight matrix is in row standardized form, then the value of S_0 is n. Comparing the standardized scores calculated by the GeoDa software, it is easy to get the conclusion that GeoDa use $\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2}$ as its principle to calculate the standard deviation. In the program, this formula will be accepted when calculating the standardized scores.

So formula (7) also is expressed as:

$$m_2=\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i-\overline{x}}{\delta}\right)^2=\frac{1}{n}\frac{\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2}{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2}=\frac{n-1}{n} \tag{8}$$

And the proportion is expressed as:

$$\gamma=S_0m_2=n\frac{n-1}{n}=n-1 \tag{9}$$

Similar with the Global Moran's I, the process of computing the Local Moran's I statistic should contain the calculation of z-score and p-value. According to the principles outlined by Cliff and Ord, the moments for I_i under the assumption of the spatial randomness are derived. For a randomization hypothesis, the expected value turns out to be [15]

$$E\left[I_i\right]=-\frac{w_i}{n-1} \tag{10}$$

and the variance is found as [15]

$$Var\left[I_i\right]=\frac{w_{i(2)}\left(n-b_2\right)}{n-1}+\frac{2w_{i(kh)}\left(2b_2-n\right)}{\left(n-1\right)\left(n-2\right)}-\frac{w_i^2}{\left(n-1\right)^2} \tag{11}$$

With $w_i=\sum_{j=1}^{n}w_{ij}$, $w_{i(2)}=\sum_{j=1}^{n}w_{ij}^2$, $2w_{i(kh)}=\sum_{k=1}^{n}\sum_{h=1,k\neq h}^{n}w_{ik}w_{ih}$, $m_2=\frac{1}{n}\sum_{i=1}^{n}z_i^2$ as the second moment, $m_4=\frac{1}{n}\sum_{i=1}^{n}z_i^4$ as the fourth moment, and $b_2=\frac{m_4}{m_2^2}$.

Cliff and Ord pointed out that "when the distribution function is unknown, we may consider the set of n random permutations, which we refer to as assumption R".[23] They also give us principles for the moments for I_i under the null hypothesis of no spatial association. There are some important principles which are useful for us to derive the expected value and variance value of each object. The observed values z_i have expectations:

$$E\left[z_i\right] = \frac{1}{n}z_1 + \frac{1}{n}z_2 + ... + \frac{1}{n}z_n = \frac{1}{n}\sum_{i=1}^{n} z_i = \overline{z} \tag{12}$$

It is zero by definition.

$$E\left[z_i^{\ 2}\right] = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2 = m_2' \tag{13}$$

$$E\left[z_i z_j\right] = -\frac{m_2'}{n-1} \tag{14}$$

$$E\left[z_i^{\ 2} z_j^{\ 2}\right] = \frac{n m_2'^2 - m_4'}{n-1} \tag{15}$$

$$E\left[z_i^{\ 2} z_j z_k\right] = \frac{2m_4' - n m_2'^2}{\left(n-1\right)\left(n-2\right)} \tag{16}$$

where $m_2' = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2$, $m_4' = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^4$ and $z_i = x_i - \overline{x}$.

Then we do the formula derivation by ourselves:

$$E\left[I_i\right] = E\left[\frac{z_i}{m_2'}\sum_{j=1}^{n} w_{ij} z_j\right] = \frac{1}{m_2'}\sum_{j=1}^{n} w_{ij} E\left[z_i z_j\right] = \frac{1}{m_2'}\sum_{j=1}^{n} w_{ij}\left(-\frac{m_2}{n-1}\right) = -\frac{\sum_{j=1}^{n} w_{ij}}{n-1} \tag{17}$$

Formula (17) is the same as formula (3).

$$E\left[I_i^{\ 2}\right] = E\left[\frac{z_i^{\ 2}}{m_2'^2}\left(\sum_{j=1}^{n} w_{ij} z_j\right)^2\right] = \frac{1}{m_2'^2} E\left[z_i^{\ 2}\left(\sum_{j=1}^{n} w_{ij}^{\ 2} z_j^{\ 2} + \sum_{k=1}^{n}\sum_{h=1,k\neq h}^{n} w_{ik} w_{ih} z_k z_h\right)\right] \tag{18}$$

$$E\left[I_i^{\ 2}\right] = \frac{1}{m_2'^2}\left(\sum_{j=1}^{n} w_{ij}^{\ 2} E\left[z_i^{\ 2} z_j^{\ 2}\right] + \sum_{k=1}^{n}\sum_{h=1,k\neq h}^{n} w_{ik} w_{ih} E\left[z_i^{\ 2} z_k z_h\right]\right) \tag{19}$$

$$E\left[I_i^{\ 2}\right] = \frac{\sum_{j=1}^{n} w_{ij}^{\ 2}}{m_2'^2} E\left[z_i^{\ 2} z_j^{\ 2}\right] + \frac{\sum_{k=1}^{n}\sum_{h=1,k\neq h}^{n} w_{ik} w_{ih}}{m_2'^2} E\left[z_i^{\ 2} z_k z_h\right] \tag{20}$$

From formula (15), (16) and (20), we get the formula (21). Then simplify the formula (21), we get the formula (22).

$$E\left[I_i^2\right]=\frac{\sum_{j=1}^{n}w_{ij}^2}{m_2'^2}\frac{nm_2'^2-m_4'}{n-1}+\frac{\sum_{k=1}^{n}\sum_{h=1,k\neq h}^{n}w_{ik}w_{ih}}{m_2'^2}\frac{2m_4'-nm_2'^2}{(n-1)(n-2)} \tag{21}$$

$$E\left[I_i^2\right]=\frac{\sum_{j=1}^{n}w_{ij}^2\left(n-\frac{m_4'}{m_2'^2}\right)}{n-1}+\frac{\sum_{k=1}^{n}\sum_{h=1,k\neq h}^{n}w_{ik}w_{ih}\left(2\frac{m_4'}{m_2'^2}-n\right)}{(n-1)(n-2)} \tag{22}$$

The variance of I_i is expressed as:

$$Var\left[I_i\right]=E[I_i^2]-\left(E\left[I_i\right]\right)^2 \tag{23}$$

Combine formula (17), (22), and (23), it is clear that we get the same formula as formula (11) which is pointed out by Anselin in 1995.

Here we must illustrate a few parameters: $m_2'=\frac{1}{n}\sum_{i=1}^{n}(x_i-\overline{x})^2$; $m_4'=\frac{1}{n}\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^4$; $m_2=\frac{1}{n}\sum_{i=1}^{n}(\frac{x_i-\overline{x}}{\delta})^2$ and $m_4=\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i-\overline{x}}{\delta}\right)^4$. But we get that $\frac{m_4'}{m_2'^2}=\frac{m_4}{m_2^2}$. So we still have the same formula as formula (11).

The z-score of I_i is expressed as:

$$z_{I_i}=\frac{I_i-E\left[I_i\right]}{\sqrt{Var\left[I_i\right]}} \tag{24}$$

For a given significant level (for example 0.05), it means that the probability of z_{I_i} appears between -1.96 and +1.96 is 95% (which equals to 1-0.05). Any z_{I_i} falls outside this range indicates that we can reject the null hypothesis at location i.

According to the z-score z_{I_i}, spatial-lag, and the standardized scores z_i, the decision is made about where it is a cluster or an outlier. For example, with a given significant level 0.05, if the z_{I_i} falls outside of range -1.96 and +1.96, a positive z_{I_i} indicates it is a cluster and a negative z_{I_i} indicates it is an outlier. For a cluster, if the value spatial lag and standardized score are positive, then it belongs to the HH type. Positive z_{I_i} with negative spatial lag and standardized score belongs to the LL type. Negative z_{I_i}

with positive spatial lag and negative standardized score belongs to LH type. And negative z_{I_i} with negative spatial lag and positive standardized score belongs to HL type. HH, LL, LH, and HL stands for different quadrant of the Moran Scatter plot which has been mentioned in the Global Moran's I section. The Figure 3 below indicates a Moran Scatter Plot with four quadrants. More details about the four types will be mentioned in the Table 1 below.

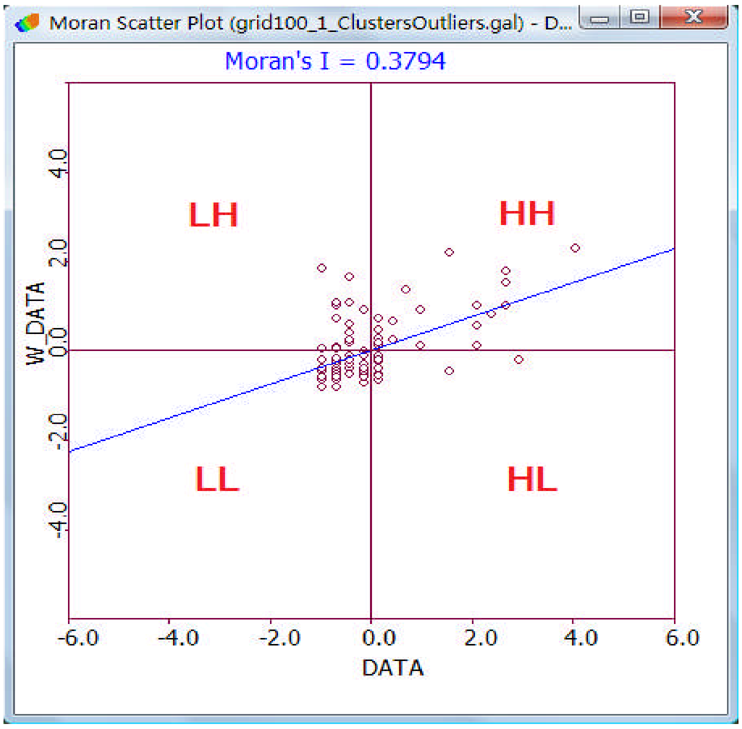


Figure 3 Moran Scatter Plot from GeoDa

Table 1 Indication of HH, LL, HL, and LH

Type	**Explanation**
HH	An object with high value surrounded with objects with high value. It is one kind of the ***cluster***.
LL	An object with low value surrounded with objects with low value. It is one kind of the ***cluster***.
HL	An object with high value surrounded with objects with low value. It is one kind of the ***outlier***.
LH	An object with low value surrounded with objects with high value. It is one kind of the ***outlier***.

3.5 Spatial Weight Matrix

The spatial weight matrix is a matrix with elements w_{ij} used to indicate the spatial relationship between object i and object j. Spatial weight matrix represents the spatial similarity. The appropriate choice of the spatial weight matrix is one of the most difficult and controversial methodological issues in exploratory spatial data analysis and spatial econometrics.[17] At least three ways exist to think about the type of spatial weights matrix that could be used in studies where spatial autocorrelation is considered: theoretical, topological, and empirical.[18] From the theoretical point of view, the matrix usually is based on distance, which is referred as distance-based spatial weight matrices. The element of the matrix is expressed as the inverse distance between any two objects, or the inverse distance squared between any two objects. From the topological point of view, the weight matrix is constructed as neighbors share the boundary, vertex or in a same area, such as contiguity weight matrix (contains Rooks case, Bishops case, and Queen's case). The empirical viewpoint is the most consistent with the intent of Cliff and Ord.[18]

In order to explain these spatial weight matrices more clearly, first we will give a sample grid dataset. Later different spatial weight matrices of this grid dataset will be shown. The number in each grid cell is the unique id of this grid cell.

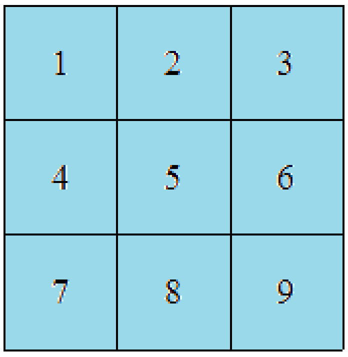


Figure 4 A sample grid dataset with unique id

The definitions of the spatial weight matrices mentioned before will be listed below.

3.5.1 Distance-based Spatial Weight Matrix

Distance-based Spatial Weight Matrix is defined as:

$$w_{ij} = \frac{1}{d_{ij}{}^{\alpha}}$$

d_{ij} is the distance between location i and location j, α is a constant value. When $\alpha = 1$, it is an inverse-based spatial weight matrix, when $\alpha = 2$, it is an inverse distance squared weight matrix.

From the definition, we get a conclusion: the smaller d_{ij} is the larger w_{ij} will be. This means when the objects are near to each other, the relationship will be closer.

Supposing $\alpha = 1$, the spatial weight matrix of Figure 4 should be:

$$W = \begin{bmatrix} 0 & 1.000 & 0.500 & 1.000 & 0.707 & 0.447 & 0.500 & 0.447 & 0.354 \\ 1.000 & 0 & 1.000 & 0.707 & 1.000 & 0.707 & 0.447 & 0.500 & 0.447 \\ 0.500 & 1.000 & 0 & 0.447 & 0.707 & 1.000 & 0.354 & 0.447 & 0.500 \\ 1.000 & 0.707 & 0.447 & 0 & 1.000 & 0.500 & 1.000 & 0.707 & 0.447 \\ 0.707 & 1.000 & 0.707 & 1.000 & 0 & 1.000 & 0.707 & 1.000 & 0.707 \\ 0.447 & 0.707 & 1.000 & 0.500 & 1.000 & 0 & 0.447 & 0.707 & 1.000 \\ 0.500 & 0.447 & 0.354 & 1.000 & 0.707 & 0.447 & 0 & 1.000 & 0.500 \\ 0.447 & 0.500 & 0.447 & 0.707 & 1.000 & 0.707 & 1.000 & 0 & 1.000 \\ 0.354 & 0.447 & 0.500 & 0.447 & 0.707 & 1.000 & 0.500 & 1.000 & 0 \end{bmatrix}$$

3.5.2 Contiguity Spatial Weight Matrix

According to GeoDa, contiguity refers to what objects are selected as neighbors. There are three different contiguity spatial weight matrices: the Rooks case; Bishops case; and Queen's case. The differences between Rooks case, Bishops case, and Queen's case are shown in Figure 5. In Figure 5 there are three graphs stand for different cases we mentioned; and the blue grid cells are the neighbors of the red cell.

From the graph, we see the differences of these three cases. The Rooks case considers the objects sharing boundaries as neighbors. The Bishops case considers the objects have same vertex as neighbors. The Queen's case considers the objects sharing boundaries and vertexes as neighbors.

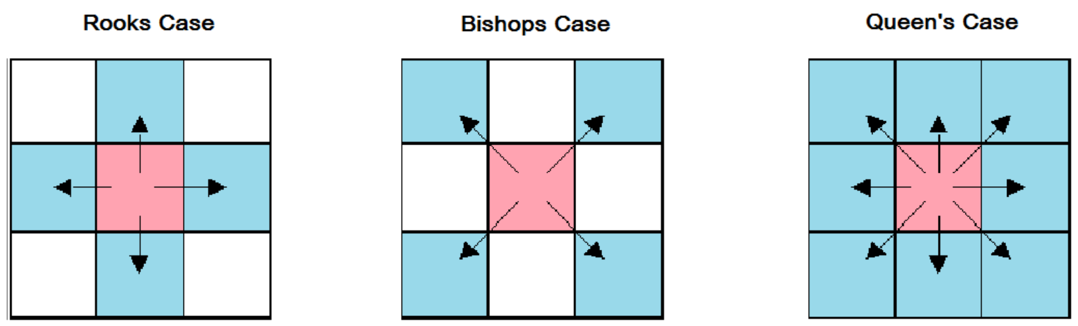


Figure 5 Three different contiguity spatial weight matrices

In some special cases, only objects under or above are considered as neighbors. Similarly, in other cases, only objects on left side or right side are considered as neighbors. A Rooks Contiguity Spatial Weight Matrix is expressed as:

$$W = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

3.5.3 Cliff and Ord's Spatial Weight Matrix

The suggestions by Cliff and Ord consist of using a combination of distance measure and the relative length of the boundary between objects. The Cliff and Ord's Spatial Weight Matrix is defined as:

$$w_{ij} = \frac{\beta_{ij}{}^{b}}{d_{ij}{}^{a}}$$

d_{ij} is the distance between location i and location j, β_{ij} is the proportion of the length of shared boundary of object i and object j to the length of the boundary of object i, a and b are two constant values defined by users.

$$W = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

The spatial weight matrix above is the Cliff and Ord's Spatial Weight Matrix of Figure 4, with $a = 1, b = 1$.

3.5.4 Dacey Spatial Weight Matrix

Similarly, Dacey defined the spatial weight matrix by taking into account the relative area of object as well as the distance measure and relative length of the boundary. The Dacey Spatial Weight Matrix is defined as:

$$w_{ij} = d_{ij}\alpha_i\beta_{ij}$$

d_{ij} is the corresponding element in contiguity spatial weight matrix with value 0 or 1, α_i is the proportion of the area of object i to the area of the whole dataset, and β_{ij} is the proportion of the length of shared boundary of object i and object j to the length of the boundary of object i. The Dacey Spatial Weight Matrix for Figure 4 is:

$$W = \begin{bmatrix} 0 & 0.028 & 0 & 0.028 & 0 & 0 & 0 & 0 & 0 \\ 0.028 & 0 & 0.028 & 0 & 0.028 & 0 & 0 & 0 & 0 \\ 0 & 0.028 & 0 & 0 & 0 & 0.028 & 0 & 0 & 0 \\ 0.028 & 0 & 0 & 0 & 0.028 & 0 & 0.028 & 0 & 0 \\ 0 & 0.028 & 0 & 0.028 & 0 & 0.028 & 0 & 0.028 & 0 \\ 0 & 0 & 0.028 & 0 & 0.028 & 0 & 0 & 0 & 0.028 \\ 0 & 0 & 0 & 0.028 & 0 & 0 & 0 & 0.028 & 0 \\ 0 & 0 & 0 & 0 & 0.028 & 0 & 0.028 & 0 & 0.028 \\ 0 & 0 & 0 & 0 & 0 & 0.028 & 0 & 0.028 & 0 \end{bmatrix}$$

3.5.5 Fixed Distance Band Weight Matrix

The Fixed Distance Band Weight Matrix is defined as:

$$w_{ij} = 1 \qquad d_{ij} < d$$

This is spatial kind of the distance-based weight matrix, d is a constant value defined by the user. Everything within a specified critical distance is included in the analysis. Everything outside the critical distance is excluded. If $d = 1$ the fixed distance band weight matrix is the same as the rook contiguity spatial weight matrix. Here we suppose $d = 2$; then the spatial weight matrix for Figure 4 is:

$$W = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

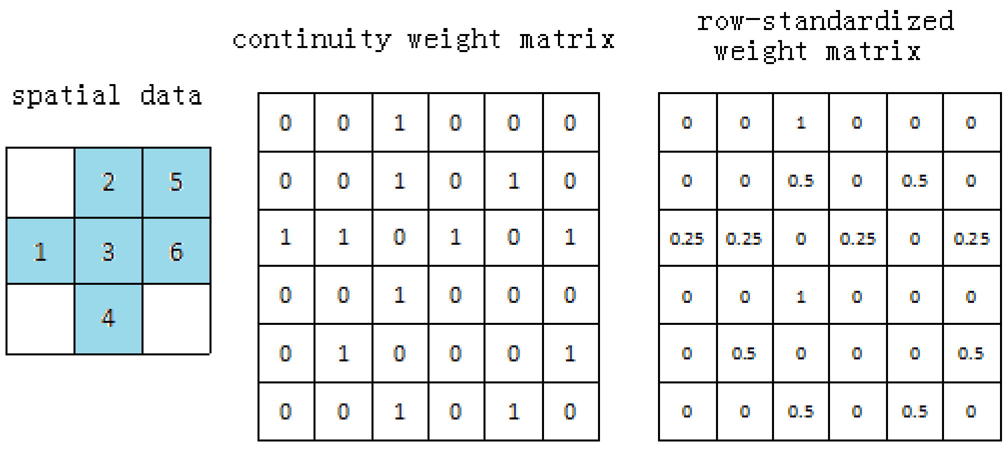
3.5.6 Other Spatial Weight Matrix

Have a look at the spatial weight matrices introduced before; it is clear that all of these spatial weight matrices are symmetric matrices. But not all the spatial weight matrices are symmetric, for example, the Row-standardized Weight Matrix which is going to be introduced is asymmetry.

(1) Row-standardized Weight Matrix

This is an important weight matrix which is used in most of the spatial analysis software. It is defined from the mathematical view, which is the sum of each row of the matrix is 1. When introducing the Local Moran's I, it is mentioned that a spatial weight matrix is in row standardized form, and this is the row-standardized weight matrix.

Figure 6 is given to indicate a row standardized form of a continuity weight matrix.



continuity weight matrix

0	0	1	0	0	0
0	0	1	0	1	0
1	1	0	1	0	1
0	0	1	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0

row-standardized weight matrix

0	0	1	0	0	0
0	0	0.5	0	0.5	0
0.25	0.25	0	0.25	0	0.25
0	0	1	0	0	0
0	0.5	0	0	0	0.5
0	0	0.5	0	0.5	0

Figure 6 Row-standardized form of continuity weight matrix

The number in each grid cell is the value of an element in spatial weight matrix. In the row-standardized form, the sum of each row equals to 1. This is the distinct characteristic of Row-standardized Weight Matrix. And it is clear that the original Continuity Weight Matrix is symmetric, but the generated row-standardized form is asymmetry.

(2) Zone of Indifference

This is the spatial weight matrix which combines the Fixed Distance Band Weight Matrix and Distance-based Spatial Weight Matrix. Anything up to a critical distance has an impact on your analysis. Once that critical distance is exceeded, the level of impact quickly drops off. The relationship between the distance and the level of impact is shown in Figure 7.

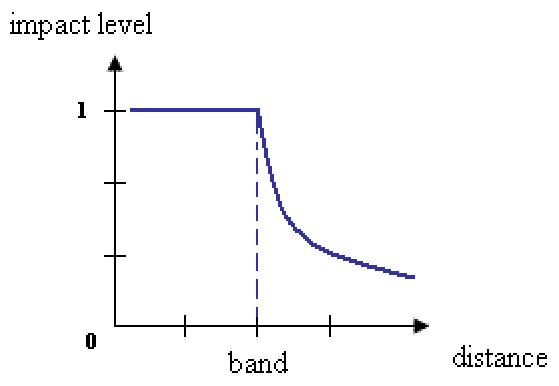


Figure 7 Zone of Indifference – distance and impact level

3.6 Summary

In this chapter, we give an introduction of Moran spatial statistic in details. First, we give the introduction of the fundamental of hypothesis tests, which will be used in calculating the z-scores of local Moran's I of each grid cell. Then, global Moran spatial statistic and local Moran spatial statistic are introduced. Finally, we describe what spatial weight matrix is, and give a list of the spatial weight matrices we have accepted in this project. In the next chapter, we will describe the implementation of the serial algorithm of how to calculate the Moran's I.

In this project, we have implemented four main spatial weight matrices: Contiguity Spatial Weight Matrix; Fixed Distance Band Weight Matrix; Distance-based Spatial Weight Matrix; Zone of Indifference. All of these spatial weight matrices will be in row standardized form.

4. Algorithm Implementation

In this section, both the serial algorithm and parallel algorithm of how to calculate Moran's I will be implemented. From last section, we get enough information about the Moran's I. First of all, we will introduce the dataset used in this project, and then analyze which parameters are needed in the calculation, next show the pipelines of the algorithms and discusses step by step.

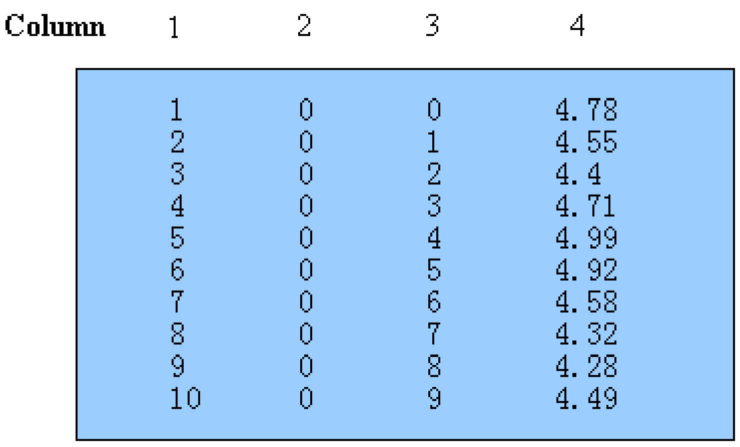
4.1 Data Generation

For spatial analysis, data is very important. No matter the data model or the data file format have influence on spatial analysis. First, we will introduce the data model used in the project, and then the spatial pattern will be represented by figures with illustration. Finally, introduce the algorithm used to generate the dataset.

4.1.1 Data Model

In GIS, there are two basic spatial data types used to represent the real world: Raster and Vector. In the raster data model, the real world is represented as a collection of single square cells. Raster data are good at representing continuous data, and the raster layers are expressed with mathematical expressions. In the vector data models, objects in the world are represented as points, lines, polygons, TINs and so on. Vector data are good at accurately representing true shape and size and representing non-continuous data. Also vector data always is used to create aesthetically pleasing maps. Figures below show the differences between raster data model and vector data model.
In this project we choose to use the raster data model as it is generated much more easily than the vector data model. The information of each dataset is stored in a .txt file. As for a GIS system, locations are very important, so the coordinates should be contained in this .txt file. In order to check the results, we need a unique id for each of the grid cell in the dataset. By using of this unique id, we get all the attributes value of

a specified grid cell. Consequently, we set four attributes for each dataset: a unique id of each of the grid cell; the x coordinate; the y coordinate; the attribute value of the measurement. The .txt file looks like Figure 8. More details about this .txt file are listed in the Table 2:



```
Column   1       2       3        4

         1       0       0        4.78
         2       0       1        4.55
         3       0       2        4.4
         4       0       3        4.71
         5       0       4        4.99
         6       0       5        4.92
         7       0       6        4.58
         8       0       7        4.32
         9       0       8        4.28
         10      0       9        4.49
```

Figure 8 Content of original dataset file

Table 2 List of original dataset file

Column	Attribute Name	Descriptions
1	POLY_ID	This is the unique id used to mark the grid cell in the dataset.
2	X	This is the x coordinate of the grid cell in the dataset.
3	Y	This is the y coordinate of the grid cell in the dataset.
4	DATA	This is the original attribute value of the grid cell in the dataset.

From the Figure 8, it is clear that the x and y coordinate are integers, and from this file, the size of each grid cell is 1×1, and the x and y coordinate are from 0. Of course, in the real world, the coordinate information would not look like this. Here, we use a simple definition about the coordinate, just focus on the spatial relationship between these grid cells, not the absolute position in the real world. More details about the dataset used in project will be introduced in section Tests Datasets.

4.1.2 Spatial Pattern

A spatial pattern is a perceptual structure, placement, or arrangement of objects which includes the space in between those objects. Patterns are recognized because of the arrangements. The patterns could be in a line or by a clustering of points. For spatial datasets, spatial pattern is a unique arrangement across the space.

Broadly considered, spatial pattern involves four basic types according to the spatial data used to represent the world:

- Point patterns

  A point pattern is a spatial pattern that is composed of closely arranged, somewhat organized points. It represents collections of objects which geographic locations are of primary interested, rather than any quantitative or qualitative attribute. Actually, we don't care about the shape of the objects, so we represent the objects as points.
- Linear network patterns

  A linear network pattern represents collections of linear objects which intersect with each other to form a network. It might be found on a map of roads or river networks.
- Surface patterns

  A surface pattern represents continuous measurements across the space; and there are no explicit boundaries. It will be found on a three-dimensional dataset, where the measurements are represented by heights at each location.
- Categorical map patterns

  A categorical map pattern represents data in which the system property of interest is represented as a mosaic of discrete patches. In this project, the grid datasets used are considered as this type.

In 1970, Waldo Tobler advanced the first law of geography "Everything is related to everything else, but near things are more related than distant things".[24] The first law of geography forms the theoretical principle for spatial analysis. If we just use the traditional statistical methods, spatial patterns cannot be indicated, Figure 9 shows such an example:

In Figure 9, there are three different grid datasets which have same traditional statistical values such as: sum; mean value; maximum value; standard deviation; median value; and so on. If we just use the traditional statistical methods, we cannot indicate the differences between these datasets.

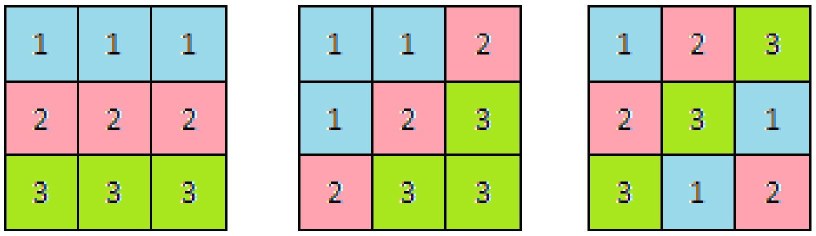
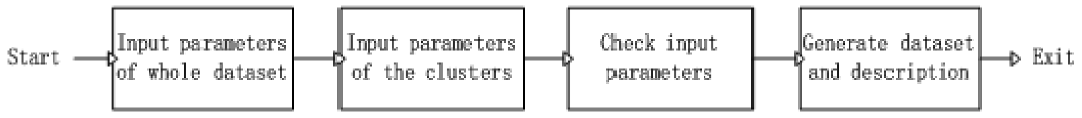


Figure 9 Datasets with different spatial patterns

From the figure, the grid cells with similar values in the first two datasets seem to be close to each other, but grid cells in the third datasets seem to be around with grid cells with different values. There are no pairs of grid cells sharing same boundary have same attribute value in the third datasets. But these pairs exist in the other two datasets. By using of Moran's I (global version or local version), the differences between these datasets will be indicated.

4.1.3 Algorithm

In this project, raster data model is chosen. For a raster dataset, there is some information we need to know about: the number of the columns; the number of the rows; the x-range and y-range of the dataset; and the attribute value for each of the grid cell. Also, some more details about the spatial pattern are needed, such as the type of the spatial pattern; information about the cluster; and the range of the attribute values in different regions. The pipeline of this program is shown in Figure 10.



Start
Input parameters of whole dataset
Input parameters of the clusters
Check input parameters
Generate dataset and description
Exit

Figure 10 Data generation schema

First of all, input parameters of the whole dataset, such as define the size of the raster dataset, specify the number of columns and rows; input the range of the attribute values for the random dataset. Next input the number of the clusters. If it is 0, then it means that the dataset is random. If it is larger than 0, then it means several clusters are included in the dataset. For the second case, we need to input the details about the clusters, such as the location and the range of attribute values for each cluster. There will be two .txt files, one named "data.txt" which stores the information needed for the calculation, and another named "info.txt" stores the description about the dataset. The files look like figure below.

If the dataset has none clusters, then it means the attribute values distribute randomly. In this case, we only need to give the range of the attribute values. By the use of ***rand()*** function provided by C++, the program generates a serial of random attribute values in the given range. Also, we need to specify the seed before using the ***rand()*** function, or the distribution of the random attribute values generated by the program at different time will be definitely the same with each other. The seed is specified by the ***srand()*** function.

If the dataset has clusters, then the grid cells belong to the clusters will have a random attribute value within the given range of attribute values for the clusters, and the grid cells don't belong to any cluster will have a random attribute value within the given range of the attribute values for the dataset.

So in the program, we first generate the random attribute values within the given range for the dataset, and store these values in an array. Then if there is any cluster, the attribute values of the grid cells belonged to clusters will be changed according to the given range of attribute values for the clusters. Finally, if the input parameters are correct, the values stored in the array will be written into "data.txt" file, and the information about this dataset will be written into "info.txt" file.

4.1.4 Tests Datasets

For the whole project, we need to generate a serial of tests datasets first of all. As we mentioned before, the datasets with same input parameters will be different from each other by using the ***srand()*** function. So after we generate these tests datasets, we should use this serial of tests datasets for the whole project. With the same tests datasets, results of parallel programs and serial programs will be compared.

Table 3 shows the information of the tests datasets used in this project. There are three different sizes of the dataset: 50×50; 100×50; and 100×100. For each size, there are three different spatial patterns: no cluster; one high cluster; and one low cluster. The clusters defined in these datasets have the same size and shape but different locations. With Gnuplot the datasets are represented as plots.

Table 3 List of tests datasets

Grid Size	Dataset	Cluster Type	Cluster Info
50×50	data_25_0	No Cluster	Range: [1, 4]
	data_25_1	One High Cluster	High Cluster: Left Low: (0, 0) Right Top: (9, 9) Range: [4, 5]
	data_25_2	One Low Cluster	Low Cluster: Left Low: (0, 0) Right Top: (9, 9) Range: [0, 1]
100×50	data_50_0	No Cluster	Range: [1, 4]
	data_50_1	One High Cluster	High Cluster: Left Low: (40, 0) Right Top: (49, 9) Range: [4, 5]
	data_50_2	One Low Cluster	Low Cluster: Left Low: (40, 0) Right Top: (49, 9) Range: [0, 1]
100×100	data_100_0	No Cluster	Range: [1, 4]
	data_100_1	One High Cluster	High Cluster: Left Low: (45, 45) Right Top: (54, 54) Range: [4, 5]
	data_100_2	One Low Cluster	Low Cluster: Left Low: (45, 45) Right Top: (54, 54) Range: [0, 1]

4.2 Problem Description

In this project, we need to calculate the Global Moran's I for the whole dataset, and Local Moran's I for each of the grid cell in the whole dataset. The Global Moran's I is calculated from the results of Local Moran's I. The results of Local Moran's I includes the local Moran's index, the z-score of the local Moran's index, the standardized value of the original attribute value, and the spatial lag and the type it belongs to.

From Chapter Moran Spatial Statistic, we get the formulas about how to calculate such values.

The standardized value of the original attribute value is defined as:

$$z_i = \frac{(x_i - \overline{x})}{\delta} \qquad \delta = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

The spatial lag for each of the grid cell is defined as:

$$Wz_i = \sum_{j=1}^{n} w_{ij} z_j$$

The local Moran's index is defined as:

$$I_i = z_i \sum_{j=1}^{n} w_{ij} z_j \qquad z_i = \frac{(x_i - \overline{x})}{\delta}$$

The z-score of the local Moran's index is defined as:

$$z_{I_i} = \frac{I_i - E[I_i]}{\sqrt{Var[I_i]}}$$

The global Moran spatial statistic:

$$\sum_{i=1}^{n} I_i = \gamma I \qquad \gamma = n-1$$

Table 4 lists all the attributes which needed in the results. It is clear that the formulas are complex, and there are many parameters needed to be calculated during the process.

After Table 4, discussions about the attributes list in the table will be given. The discussions are about the conceptions in the calculations

Table 4 List of attributes needed to be calculated

Attribute Name	Descriptions	Formulas
I_DATA	The local Moran's index for each of the grid cell.	$I_i = z_i \sum_{j=1}^{n} w_{ij} z_j$
Z_DATA	The z-score of the local Moran's index for each of the grid cell.	$z_{I_i} = \frac{I_i - E[I_i]}{\sqrt{Var[I_i]}}$
STD_DATA	The standardized value of the original attributes value.	$z_i = \frac{(x_i - \bar{x})}{\delta}$
LAG_DATA	The spatial lag for each of the grid cell.	$Wz_i = \sum_{j=1}^{n} w_{ij} z_j$
CLU_DATA	The spatial type it belongs to.	$0: z_{I_i} \in [-1.96, 1.96]$ $1: z_{I_i} \notin [-1.96, 1.96]; z_i > 0; Wz_i > 0$ $2: z_{I_i} \notin [-1.96, 1.96]; z_i < 0; Wz_i < 0$ $3: z_{I_i} \notin [-1.96, 1.96]; z_i < 0; Wz_i > 0$ $4: z_{I_i} \notin [-1.96, 1.96]; z_i > 0; Wz_i < 0$

Next, we will analyze which parameters are needed in the calculation.

- Standardized value

  From the formulas provided in the last chapter, we see that comparing to the original attribute values, the standardized values are used in the calculation. So first of all, we should preprocess the original attribute values. After the preprocessing, the standardized values will be used in the next calculations. The standardized value of the original attribute value is defined as:

$$z_i = \frac{(x_i - \bar{x})}{\delta} \qquad \delta = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

- Spatial weight matrix

  As we mentioned before, there are different types of spatial weight matrix. With different spatial weight matrices, the results are different. So first of all, we need to generate the spatial weight matrix for the dataset. For a specified grid cell i, we get its spatial weights with other grid cells by calculating the distances between other grid cells and making decision under the criterion.

- Spatial lag

  According to the formulas, the expression of spatial lag is expressed as:

  $$Wz_i = \sum_{j=1}^{n} w_{ij} z_j$$

  z_i is the standardized score of the attribute value, and w_{ij} is the corresponding spatial weight between object i and j. This calculation should be operated for each of the grid cell in the dataset. This spatial lag is used in calculating the local Moran's I.

- Local Moran's I

  The formula of calculating the local Moran's I is expressed as:

  $$I_i = z_i \sum_{j=1}^{n} w_{ij} z_j$$

  Combing with the spatial lag, this formula is also expressed as:

  $$I_i = z_i W z_i$$

  If we have already had the value of spatial lag, we get the value of local Moran's I by operating multiplication for once. So we get spatial lag and local Moran's I in the same loop structure.

- Z-score of the local Moran's I

  This is the most complex step during the calculation. With tests, we get the conclusion that this is the most time-consuming part of the program. So we need to pay more attention in this part. In order to get the correct results, some parameters are needed, such as: b_2; w_i; $w_{i(2)}$; $2w_{i(kh)}$. The expressions of these parameters are list:

  $$b_2 = \frac{m_4}{m_2^{\ 2}} \qquad m_2 = \frac{1}{n}\sum_{i=1}^{n} z_i^{\ 2} \qquad m_4 = \frac{1}{n}\sum_{i=1}^{n} z_i^{\ 4}$$

  As for the same dataset, there is only one value of b_2, so we put it outside of the loop structure.

  $$w_i = \sum_{j=1}^{n} w_{ij}$$

$$w_{i(2)} = \sum_{j=1}^{n} w_{ij}^{\ 2}$$

w_i and $w_{i(2)}$ are parameters about the spatial weight matrix only, we get these two parameters in the same loop structure.

$$2w_{i(kh)} = \sum_{k=1}^{n} \sum_{h=1,k\neq h}^{n} w_{ik} w_{ih}$$

Comparing $2w_{i(kh)}$ with w_i and $w_{i(2)}$, we know that $2w_{i(kh)}$ needs a loop structure with nearly n^2 times, but w_i and $w_{i(2)}$ only need a loop structure with n times. So if we use an efficient loop structure in this part, the performance will be improved significantly. In section Spatial Weight Matrix, we have mentioned that, for most of the spatial weight matrices, they are sparse matrix, which means that most of the elements in the matrix are zeros. Using this characteristic, we use shells (if-else) to save the computing time. Comparing the programs with shells with programs without shells, we get the conclusion that shells can help saving the computing time and improving the performance. More details are shown in the programs.

4.3 Serial Algorithm

So far, we have introduced the dataset used in this project, and then analyzed parameters needed in the calculation. Now it is time to implement the serial algorithm. The algorithmic scheme of our approach to get these values is presented in Figure 11.

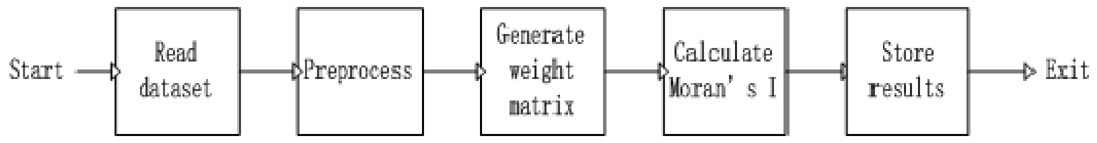
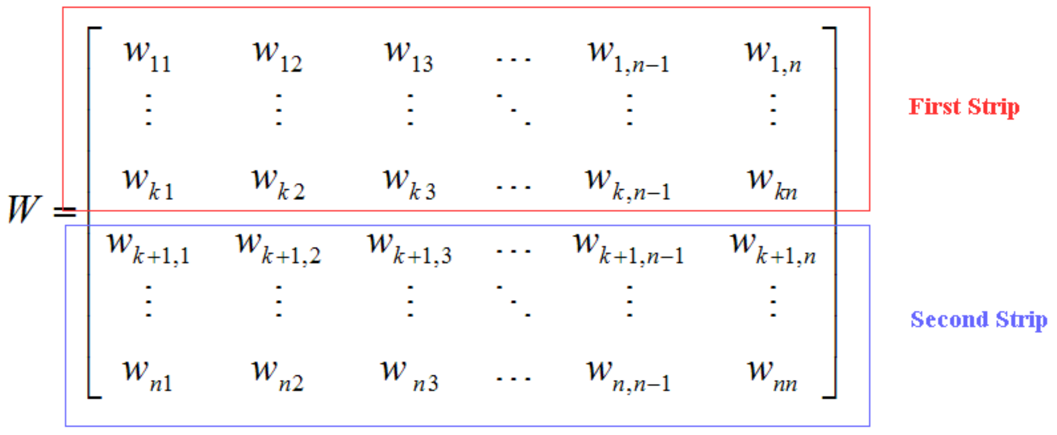


Figure 11 Serial algorithm schema

First, read the information of the grid dataset from a data.txt file, and stores the information in the memory. Second, pre-process the source dataset, and calculate the standardized values of the original attribute values. Third, generate a spatial weight matrix of this dataset according to the user’s choice.(There are four different spatial weigh matrices) Then, use methods to calculate the spatial lag, local Moran’s index, and z-score of local Moran’s index for each of the grid cell. Finally, get the value of the global Moran’s index by combining the results of the local Moran’s indices.

When trying to implement the serial algorithm, we found that for the dataset with many grid cells there may be not enough memory for the computer to store the data needed in the calculation. This is because the calculation of Moran’s I is computing intensive and memory intensive. In order to solve this problem, a solution is given in the serial algorithm.

The problem is that there is not enough memory to store the data in the computing, especially the spatial weight matrix data. So we consider not store the whole spatial weight matrix in the memory. First partition the spatial weight matrix into parts, and compute one part at a time. Then the memory will be enough for each part, but more time will be used for the computing. This solution is practicable, because for each grid cell, no matter the local Moran’s I or the z-score of the local Moran’s I, the calculation is related to one row in the spatial weight matrix. So it is reasonable to partition the spatial weight matrix into strips. The separation of the spatial weight matrix is shown in Figure 12. Also this solution is used in the implementation of the parallel algorithm.



$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \cdots & w_{1,n-1} & w_{1,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{k1} & w_{k2} & w_{k3} & \cdots & w_{k,n-1} & w_{kn} \\ w_{k+1,1} & w_{k+1,2} & w_{k+1,3} & \cdots & w_{k+1,n-1} & w_{k+1,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{n1} & w_{n2} & w_{n3} & \cdots & w_{n,n-1} & w_{nn} \end{bmatrix}$$

Figure 12 Separation of spatial weight matrix in serial algorithm

4.4 Parallel Algorithm

The aim of parallel computing is to save time for a computation by using multiple processors. Three fundamental problems must be solved when developing a parallel algorithm: identification of parallelism; partitioning of the problem into a set of sequential tasks; and scheduling the tasks.[6] When writing parallel programs we should choose to use the functional parallelism or data parallelism.

- Functional parallelism is one decomposes a complex task into simpler sub-tasks which sometimes be overlapped in the same way as process pipelines.
- Data parallelism is one can partition the data and regard each sub-domain as a task which is performed in parallel.

So first of all, a decision about which parallelism is suitable for the problem must be done. In this section we will first compare the functional parallelism and data parallelism, analyze the advantages and disadvantages of both the parallelisms, and finally give a decision about the type of the parallelism used for the parallel algorithm.

4.4.1 Functional Parallelism

Considering the functional parallelism, the task should be divided into sub-tasks, and each task has its own functions. These sub-tasks can be parallel or serial. The perfect situation is that all the sub-tasks run at the same time without waiting for the results of other sub-tasks, and the time for each sub-task is almost the same. But this is the perfect situation, for most of the cases; this objective cannot be achieved.

$$\delta = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$z_i = \frac{(x_i - \bar{x})}{\delta}$$

$$m_2 = \frac{1}{n}\sum_{i=1}^{n} z_i^{\,2}$$

$$m_4 = \frac{1}{n}\sum_{i=1}^{n} z_i^{\,4}$$

$$Wz_i = \sum_{j=1}^{n} w_{ij} z_j$$

$$w_i = \sum_{j=1}^{n} w_{ij}$$

$$w_{i(2)} = \sum_{j=1}^{n} w_{ij}^{\,2}$$

$$2w_{i(kh)} = \sum_{k=1}^{n}\sum_{h=1,k\neq h}^{n} w_{ik} w_{ih}$$

Figure 13 Parameters to be calculated

So for a functional parallelism, we need to try to find out the parallel processes, and make sure these processes are finished almost in the same time. This solution is referred as functional decomposition. In this approach, the focus is on the computation that is to be performed rather than on the data manipulated by the computation. The problem is decomposed according to the work that must be done. Each task then performs a portion of the overall work.

Figure 13 shows the parameters needed to be calculated. Each parameter shows as distinct entity. Next we examined functional dependencies between these parameters. These dependencies are summarized in the following two categories:

- Compute parameters related to the attributes values.
- Compute parameters related to the spatial weight matrix.

Next the dependencies between these parameters will be presented in the form of a dependency graph. From Figure 14, it is clear that, although both of the categories are executed independently, but some parameters are computed parallel, some must be computed sequentially.

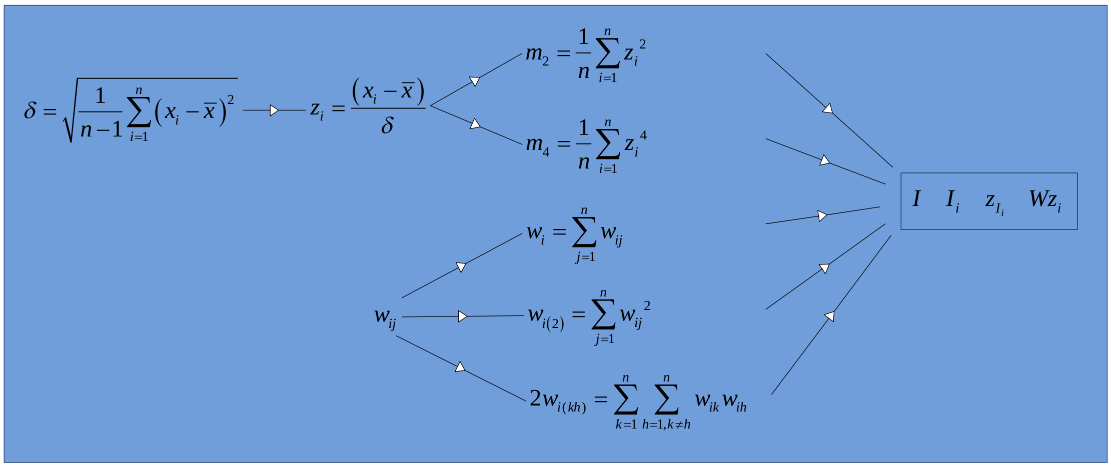

$\delta = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$
$z_i = \frac{(x_i - \bar{x})}{\delta}$
$m_2 = \frac{1}{n}\sum_{i=1}^{n} z_i^2$
$m_4 = \frac{1}{n}\sum_{i=1}^{n} z_i^4$
$w_i = \sum_{j=1}^{n} w_{ij}$
w_{ij}
$w_{i(2)} = \sum_{j=1}^{n} w_{ij}^2$
$2w_{i(kh)} = \sum_{k=1}^{n}\sum_{h=1,k\neq h}^{n} w_{ik}w_{ih}$
$I \quad I_i \quad z_{I_i} \quad Wz_i$

Figure 14 Dependency graph presents the dependencies between parameters

Also in Section Parallel Algorithm, we have said that some parameters are much more complex and time-consuming. Especially the $2w_{i(kh)}$ parameter, most of the time is spent in this section. So if functional parallelism is taken as the solution for parallel algorithm, it is hard to achieve the load balance. Next consider the data parallelism solution.

4.4.2 Data Parallelism

Considering the data parallelism is one method partite the data into parts, and performs the same process on each part. The processes are performed in parallel. But for a data parallelism, we need to make sure that the sub-datasets are independent on each other, or there is little relationship between these sub-datasets. We know that many algorithms have data dependencies, and this has a bad influence in parallelization. The solution for the data parallelism is referred as domain decomposition. Data used in the problem will be partitioned, and task will be performed on one of the subset of the data. Among the various parallel computing algorithms, domain decomposition method has become much popularity in the recent years.[25] For a two-dimension domain, there are two basic domain decomposition methods: domain decomposition into strips and domain decomposition into squares.

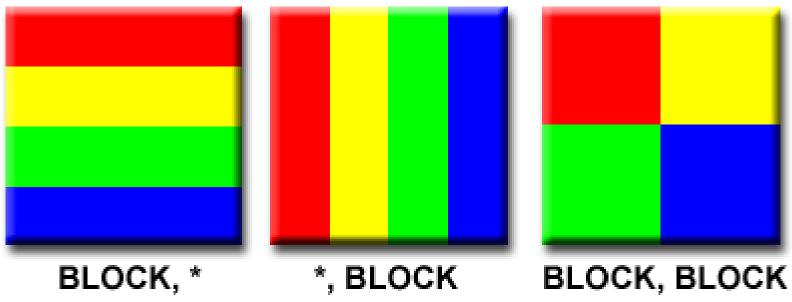


Figure 15 Domain decomposition methods for two-dimension domain

The calculation of Moran’s I can be considered as a serial of matrix operations. Formulas listed before has a matrix form. Here we take the calculation of the spatial lag for an example.

The matrix form of this formula is expressed as:

$$Wz = W \times Z$$

Here Wz is a matrix with n rows and one column, W is a matrix with n rows and n columns, and Z is another matrix with n rows and one column. Have a look at the matrices listed below.

$$Wz = \begin{bmatrix} Wz_1 \\ Wz_2 \\ \vdots \\ Wz_n \end{bmatrix} \qquad W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix} \qquad Z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$$

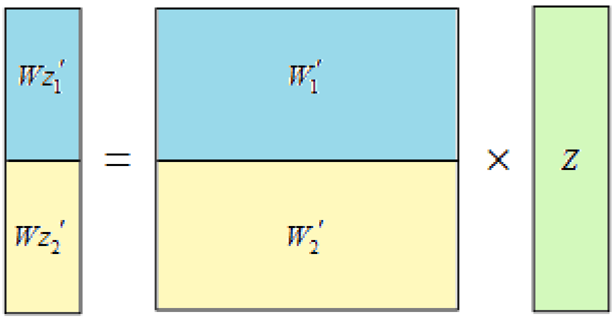
In this case, we consider the problem as a serial of matrix operations on a few matrices. How to use decomposition method to solve this problem will be discussed next.

First, the matrix W is the most complex matrix which is divided into strips and squares as we mentioned before. Now we will analyze which method is better.

If we divide the matrix into strips, for example two strips, and then the formula is expressed as:

$$\begin{bmatrix} Wz_1' \\ Wz_2' \end{bmatrix} = \begin{bmatrix} W_1' \\ W_2' \end{bmatrix} \times \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} \qquad Wz_i' = \begin{bmatrix} Wz_j \\ \vdots \\ Wz_k \end{bmatrix} \qquad W_i' = \begin{bmatrix} w_{j1} & w_{j2} & \cdots & w_{jn} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{kn} \end{bmatrix}$$

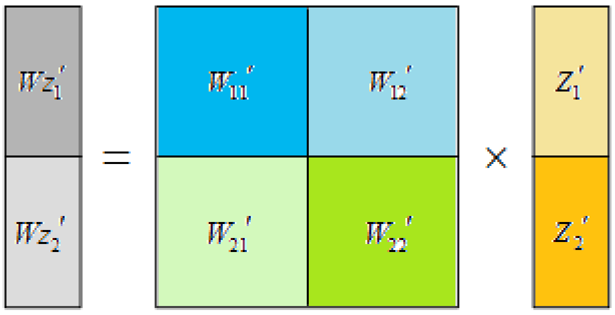
The matrix Wz_i' and W_i' are part of the original matrix Wz and W. Or see the graph below. Notice that $Wz_i' = W_i' \times Z$, and i is the id of the script. See that the calculations for each strip are the same and independent with each other. This means that each processor takes charge of one strip, and gets the result for this strip without communications with other processors. The only thing needed by processor is original dataset. All the processors work in parallel.



If we divide the matrix into squares, for example four squares, and then the formula is expressed as:

$$\begin{bmatrix} Wz_1' \\ Wz_2' \end{bmatrix} = \begin{bmatrix} W_{11}' & W_{12}' \\ W_{21}' & W_{22}' \end{bmatrix} \times \begin{bmatrix} Z_1' \\ Z_2' \end{bmatrix} \qquad Wz_i' = \begin{bmatrix} Wz_j \\ \vdots \\ Wz_k \end{bmatrix} \qquad W_{ij}' = \begin{bmatrix} w_{ml} & \cdots & w_{mk} \\ \vdots & \ddots & \vdots \\ w_{nl} & \cdots & w_{nk} \end{bmatrix} \qquad Z_i' = \begin{bmatrix} z_m \\ \vdots \\ z_n \end{bmatrix}$$

We also use a graph to show how the division likes like.



And in this kind of division, the calculation would be more complex than the first one. Because the calculation for each part should be expressed as:

$$Wz_i' = W_{i1}' \times Z_1' + W_{i2}' \times Z_2'$$

In summary, the first data decomposition should be better than the second one. There are several reasons: first, the calculation for each part might be much easier if we take the first decomposition method; second, considering the situation that the number of the sub-dataset cannot be divided by the number of the rows (or columns) with no remainder, the calculation for each square could be different; third, the calculation of spatial lag is just one parameter in the computation, other parameters are more difficult to be calculated by the second method. For example $2w_{i(kh)} = \sum_{k=1}^{n} \sum_{h=1,k\neq h}^{n} w_{ik} w_{ih}$ would be difficult to calculate if we divide the spatial weight matrix into square.

4.4.3 Solution

According to the exposition given before, we think it is an advisable decision to use domain decomposition which partitions the spatial weight matrix into scripts in rows. In this section, we will describe how the parallel algorithm is implemented in details. According to an online book named “Designing and Building Parallel Programs” which is provided by Ian Foster, there are four parallel programming tools in wide use:

Compositional C++; Fortran M; High Performance Fortran; and Message Passing Interface (MPI).

Here we choose to use MPI. MPI is a particular message passing library, which is proposed by a committee of vendors, implementers, and users. MPI supports two basic types of communication: point to point communication; and collective communication. The point to point communication is like two processors, one is a sender and the other is the receiver, and the communication is between two processors only. The collective communication means more than two processors sharing same data. There are five kinds of collective communications: barrier synchronization; broadcast; scatter; gather; and reduction. MPI supports two kinds of bindings: one is for C++, the other is for FORTRAN. C++ bindings are used in this project.

As we mentioned before, domain decomposition will be used to partition the spatial weight matrix into scripts in rows. With this method, the calculations for each strip are done with different processors in parallel. Each processor takes charge of one strip, and gets the result for this strip without communications with other processors. The only thing needed by each processor is original dataset. So broadcast and gather which belongs to collective communication will be used. Figure 16 shows how broadcast and gather communication works.

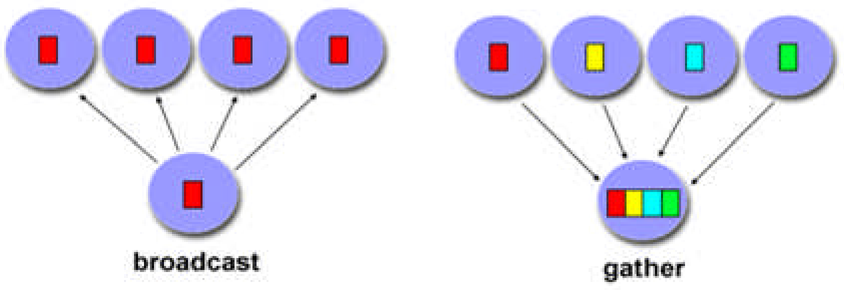


Figure 16 Collective communication: broadcast and gather

First, the root processor will broadcast the original dataset which contains grid ids, x coordinates, y coordinates, and attribute values of the whole dataset to all the processors in use. Then each processor runs the same program in parallel. Tasks running in different processors are all the same. Finally, the result of each processor will be gathered by the root processor in order just like the figure shows.

Here we need to emphasize that the original dataset is broadcasted, not the spatial weight matrix. So each processor needs to generate the spatial weight matrix at the beginning with the dataset received. As the datasets received by each processor are the same, so the generated spatial weight matrices are the same. May be you will think that it is a waste of time, because you broadcast the spatial weight matrix, divide it and send each strip to the corresponding processors. There are two reasons we don’t do it like that: first the time used for generating the spatial weight matrix is just a few seconds; second if we generate the spatial weight matrix and divided into strips then send the messages to other processors, the operations in root processor are complex. We need to notice that, each processor receives different message, so that point to point communications are used, and the root processor needs to send the message to each processor in rank order which will take more time in communication.

In addition, the decomposition of the spatial weight matrix has already been done when we know how many processors will be used. If there are five processors in use, then the spatial weight matrix will be divided into five strips in rows. Each strip has the same number of rows. Of course, there will be the situation that the number of the rows of the spatial weight matrix cannot be divided by the number of processors with no remainder. In this case, we will add a few rows with zeros at the end of the spatial weight matrix and generate a new spatial weight matrix. Figure 18 shows operations we take when the spatial weight matrix cannot be divided without remainder.

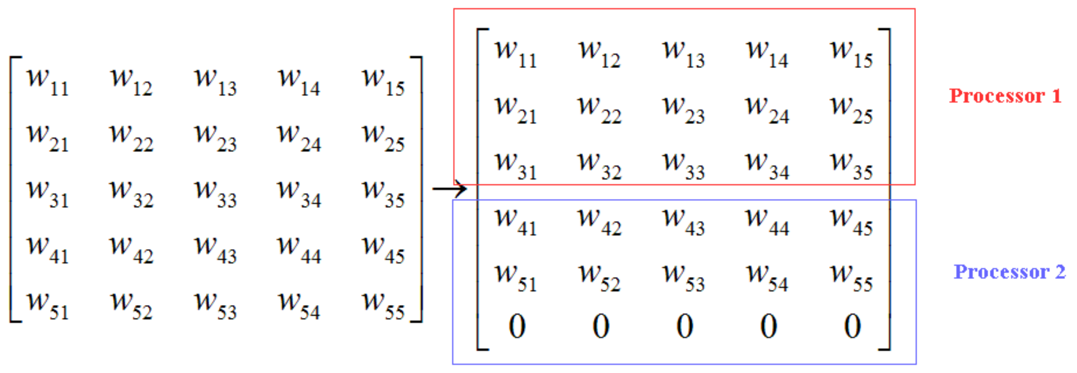


Figure 17 Special case of dividing spatial weight matrix

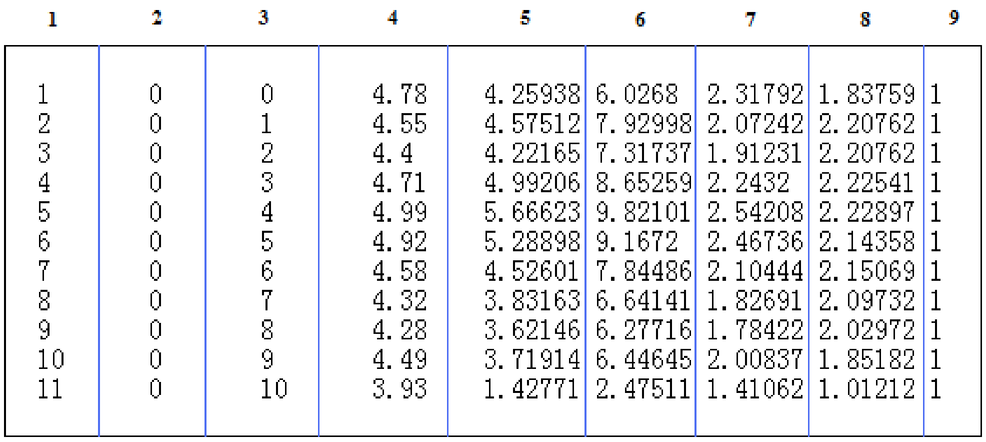
On one hand the new spatial weight matrix is divided into the strips with the same size, on the other hand the results won’t change.

There is another reason of doing this, the gather function requires all the processors send the messages with same data type as well as count. So that the root processor receives the messages and store those in rank order.

In this case, the spatial weight matrix is a matrix with five rows and five columns, and the program will run with two processors. The spatial weight matrix cannot be divided into two strips with same size, so at the end of the spatial weight matrix, a new row with zeros is added. Then the new spatial weight matrix is divided into two strips with same size. Each strip is a matrix with three rows and five columns.

4.5 Results

In order to compare the serial and parallel algorithm, the results will be shown in the same form. The results are stored in a .txt file which is named as result.txt. This file is looks like:



1	2	3	4	5	6	7	8	9
1	0	0	4.78	4.25938	6.0268	2.31792	1.83759	1
2	0	1	4.55	4.57512	7.92998	2.07242	2.20762	1
3	0	2	4.4	4.22165	7.31737	1.91231	2.20762	1
4	0	3	4.71	4.99206	8.65259	2.2432	2.22541	1
5	0	4	4.99	5.66623	9.82101	2.54208	2.22897	1
6	0	5	4.92	5.28898	9.1672	2.46736	2.14358	1
7	0	6	4.58	4.52601	7.84486	2.10444	2.15069	1
8	0	7	4.32	3.83163	6.64141	1.82691	2.09732	1
9	0	8	4.28	3.62146	6.27716	1.78422	2.02972	1
10	0	9	4.49	3.71914	6.44645	2.00837	1.85182	1
11	0	10	3.93	1.42771	2.47511	1.41062	1.01212	1

Figure 18 Content of result file

It is clear that there are nine columns which stand for nine different attributes. Table 5 shows what these attributes are. Table 5 contains the attribute name for each column as well as a shortly description of this attribute. As we mentioned before, Gnuplot is used to visualize the results so that users understand the results easier, a description of the results in Gnuplot will be introduced next.

Table 5 List of result file

Column	Attribute Name	Descriptions
1	POLY_ID	This is the unique id used to mark the grid cell in the dataset.
2	X	This is the x coordinate of the grid cell in the dataset.
3	Y	This is the y coordinate of the grid cell in the dataset.
4	DATA	This is the original attribute value of the grid cell in the dataset.
5	I_DATA	This is the local Moran's I value of the grid cell in the dataset.
6	Z_DATA	This is the z-score of the local Moran's I value of the grid cell in the dataset
7	STD_DATA	This is the standardized value of the original attribute value of the grid cell in the dataset.
8	LAG_DATA	This is the spatial lag value of the grid cell in the dataset.
9	CLU_DATA	This is the cluster type which the grid cell in the dataset belongs to.

Gnuplot is a free program which is used to plot data and functions. In this project, Gnuplot is used to show the results. For each of the test, the results will be plotted and four main plots will be shown in multiplot mode. Here we give these four different plots and explain what these plots stand for. All the four plots are from the same dataset. Here we use dataset named “data_25_1” mention in Table 3 as the sample dataset, and Contiguity Spatial Weight Matrix is used to get the results.

Gnuplot plots more than one figure in a single frame, which is referred as a “multiplot” mode. Users benefit from this mode, because they can compare different plots in a single frame.

Now we are going to introduce these four different plots.

4.5.1 Original Dataset Plot

This is a plot used to display the original attribute values. The grid cells with same colour have same attribute value. Because the original attribute values are continuous so the colours are continuous. This plot is used to check whether the results are reliable. Figure 19 shows the original dataset plot of the sample dataset.

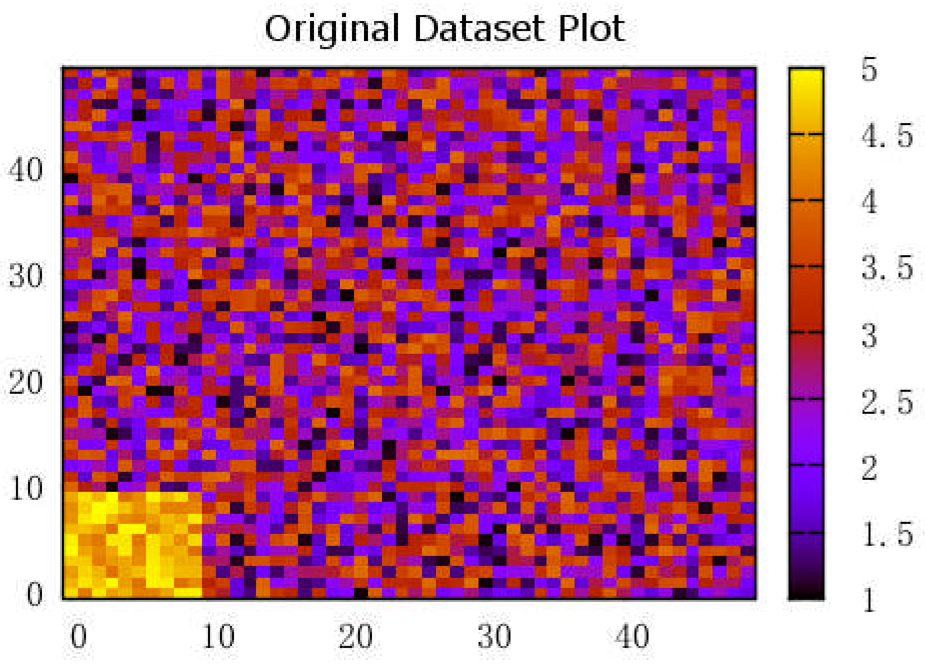


Figure 19 Original Dataset Plot

From Figure 19, it is clear that on the left bottom corner, there is a cluster with high value. According to Table 3, “data_25_1” is a dataset with a high cluster, the left low point of this cluster is (0, 0) and the right top point is (9, 9), just like the graph.

4.5.2 Moran Index Plot

This is a plot used to display the local Moran’s I values. The grid cells with same colour have same local Moran’s I value. The same as original dataset plot, the colours are continuous for the local Moran’s I value are continuous. This plot shows the range of the local Moran’s I value. And it emphasizes the areas with very high values where the attributes values in this area are similar.

From Figure 20, we see that at the same location of the cluster contained in Figure 19, grid cells have a high local Moran’s index. According to the introduction of local Moran’s I, we know that positive value means grid cell is surrounded by grid cells with similar attribute values. So it is easy to understand that the grid cells in the cluster have high local Moran’s index.

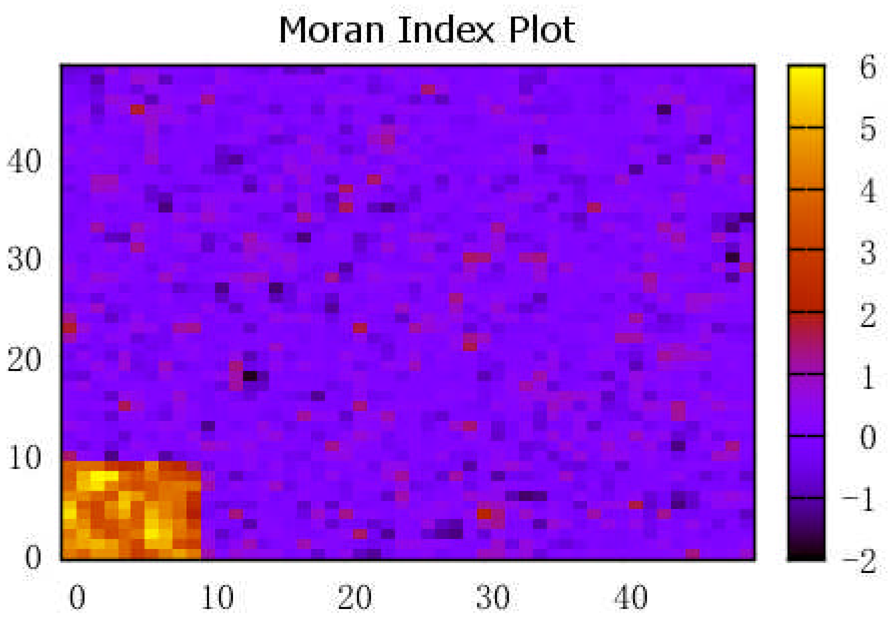


Figure 20 Moran Index Plot

4.5.3 Cluster Type Plot

This is a plot used to show the cluster type the grid cells belong to. There are only five different colours. Grey stands for no significant. Red stands for HH type. Blue stands for LL type. Yellow stands for LH type. Green stands for HL type. The results are under a given significant level (in this project the significant level is 0.05) which means when z-score lies out of a specified range (in this project the range is [-1.96, 1.96]) it is significant and need to make a decision of which type it belongs to. The rules used to make the decision are listed.

From Figure 19, we know that a high cluster exists on the left bottom corner, so that the grid cells in this corner should belong to HH type, so they are red in Cluster Type Plot, just like Figure 21.

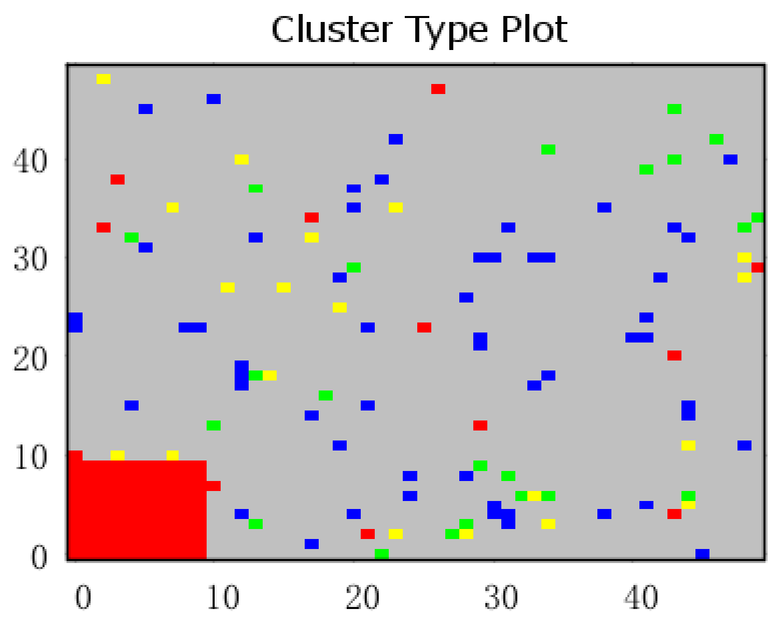


Figure 21 Cluster Type Plot

4.5.4 Moran Scatter Plot

This plot is generated with standardized value on the horizontal axis and its spatial lag on the vertical axis, see Figure 22.

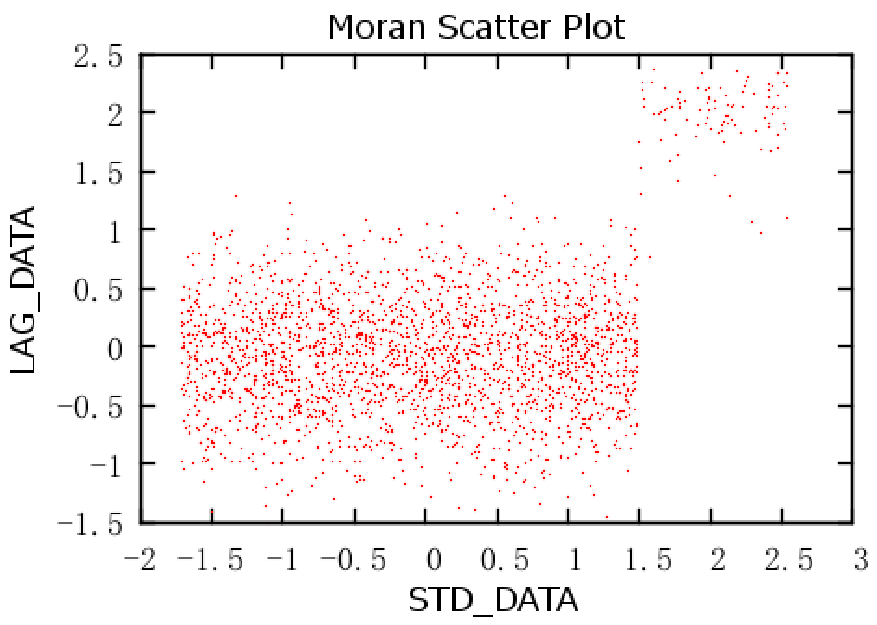


Figure 22 Moran Scatter Plot

There are four quarters in the Moran Scatter Plot. Each quarter stands for a cluster type. Of course, this should be under the specified significant level, but it cannot be shown in this plot.

For more information, the slope of the regression line through this scatter corresponds to the global Moran's I statistics. It is used to visualize the spatial autocorrelation. As it cannot show the cluster type, many commercial GIS software do not accept this plot as one of the output results.

4.6 Summary

In this section, we introduce the spatial data used in this project first of all. Then implement a program used to generate tests datasets for the programs of calculating Moran's I. Then, we mainly describe the implementations of both of serial algorithm and parallel algorithm. At the end of this project, Gnuplot is used to visualize the results which stored in a .txt file.

To make it clear, Gnuplot is not the only way to visualize the results. It is also transformed to a .dbf file, which is used in many professional GIS software. With the help of these GIS software, users also check the results directly perceived through the senses. Users choose their own methods to check and display the results.

5. Tests in Astral Environment

The key issue in the parallel computing in a single application is the speedup achieved, specially its dependence on the number of processors used. The speed-up ratio on n processors is defined as:

$$S_p = \frac{T_0}{T_p}$$

where T_0 is the time for solving the problem on a single processor with the fastest serial algorithm, and T_p is the total time for solving the problem using n processor. But it is difficult to find out T_0, so we use an approximate value T_1 instead of T_0. T_1 is the time for solving the problem using a single processor. And the speed-up ratio is expressed as:

$$\overline{S}_p = \frac{T_1}{T_p}$$

Another item related with speed-up ratio is “efficiency” which is given by:

$$E_p = 100\% \times \frac{\overline{S}_p}{n}$$

For example, the program takes 8 seconds to run on a single processor, while the parallel program takes 2 seconds running on five processors, then

$$\overline{S}_p = \frac{T_1}{T_p} = \frac{8}{2} = 4$$

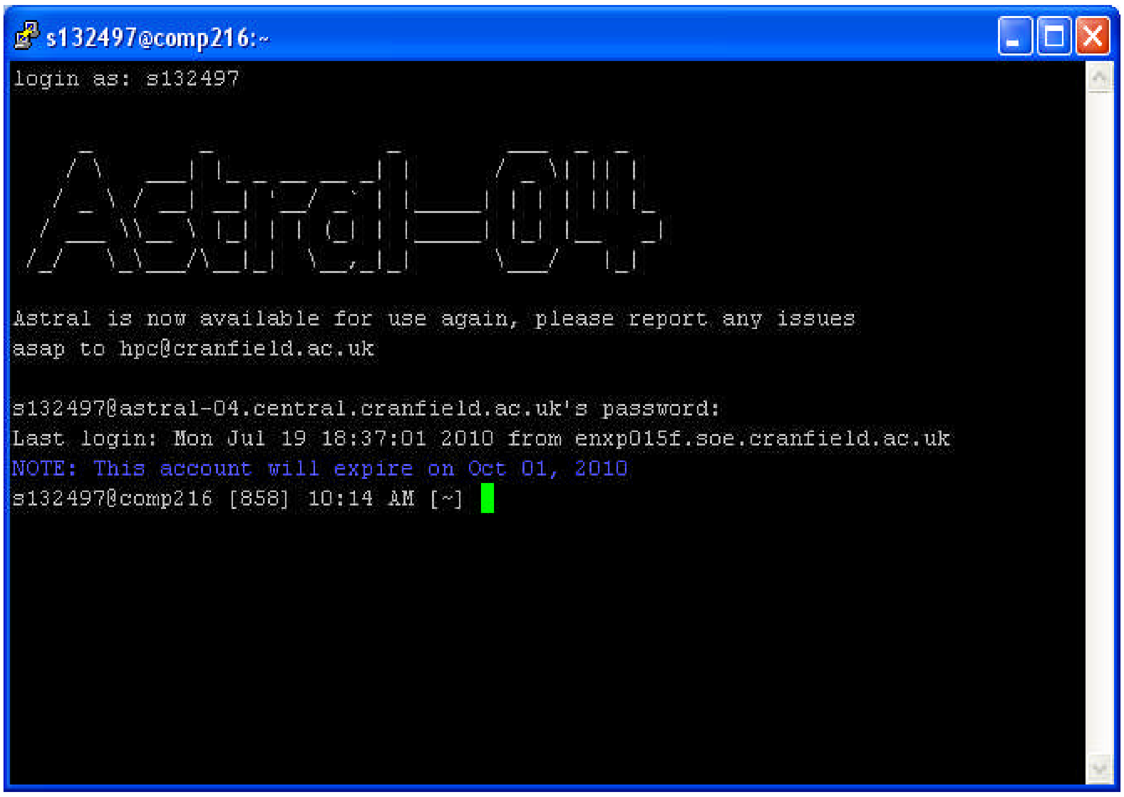
$$E_p = 100\% \times \frac{\overline{S}_p}{n} = 100\% \times \frac{4}{5} = 80\%$$

This means the parallel algorithm exhibits a speed-up ratio of 4 with five processors giving an efficiency of 80%.

The speed-up ratio and efficiency is used to define the parallel performance. In this chapter, we will describe the tests environment and how we do the experiments. A summary will be given at the end of this chapter.

5.1 Tests Environment

The parallel program developed will be tested in Astral environment. Astral is a supercomputer site built in 2007 by Cranfield University. It comprises 856 processors with the overall peak performance of 7.3 TFlops. It is accessed to with user name and password provided at the beginning of the course. Astral is accessible via Linux as well as Windows. In the project, we choose to access to Astral via Windows. Figure 23 is the graphic user interface.



```
login as: s132497

Astral is now available for use again, please report any issues
asap to hpc@cranfield.ac.uk

s132497@astral-04.central.cranfield.ac.uk's password:
Last login: Mon Jul 19 18:37:01 2010 from enxp015f.soe.cranfield.ac.uk
NOTE: This account will expire on Oct 01, 2010
s132497@comp216 [858] 10:14 AM [~]
```

Figure 23 Graphic user interface to use Astral

Next are the software and settings we need. In order to access to Astral, we need the help from Hummingbird Exceed and Putty. Exceed is a market and technology leading PC X server that empowers Windows users with cost effective access to X Window applications on UNIX and Linux hosts with unparalleled performance and strong security. Putty is a free implementation of Telnet and SSH for Windows and UNIX platforms, along with a terminal emulator. Newer version of Hummingbird Exceed can support the SSH connection. SSH or Secure Shell is a protocol for creating a secure connection between two networked computers so that data are exchanged securely. It is viewed as a secure version of FTP. X11 stands for X Window System which is a computer software system and network protocol that allows you to run software on a UNIX or Linux server in a graphical user interface so that you control the program or software running on a remote server. The secure way to do this is to forward X11 using SSH connection.

First, we need to run the "Exceed.exe" application. After that run "Putty.exe" and a window will show. We need to setup the connection info in "Host Name" field which should be "astral-04.central.cranfield.ac.uk", and select SSH which is using port 22. In the tree view control on the left of the Putty window, expand "SSH" and click "Enable X11 forwarding". Save this setting and connect to the host with given host name. Figure 24 shows the configuration window of Putty.

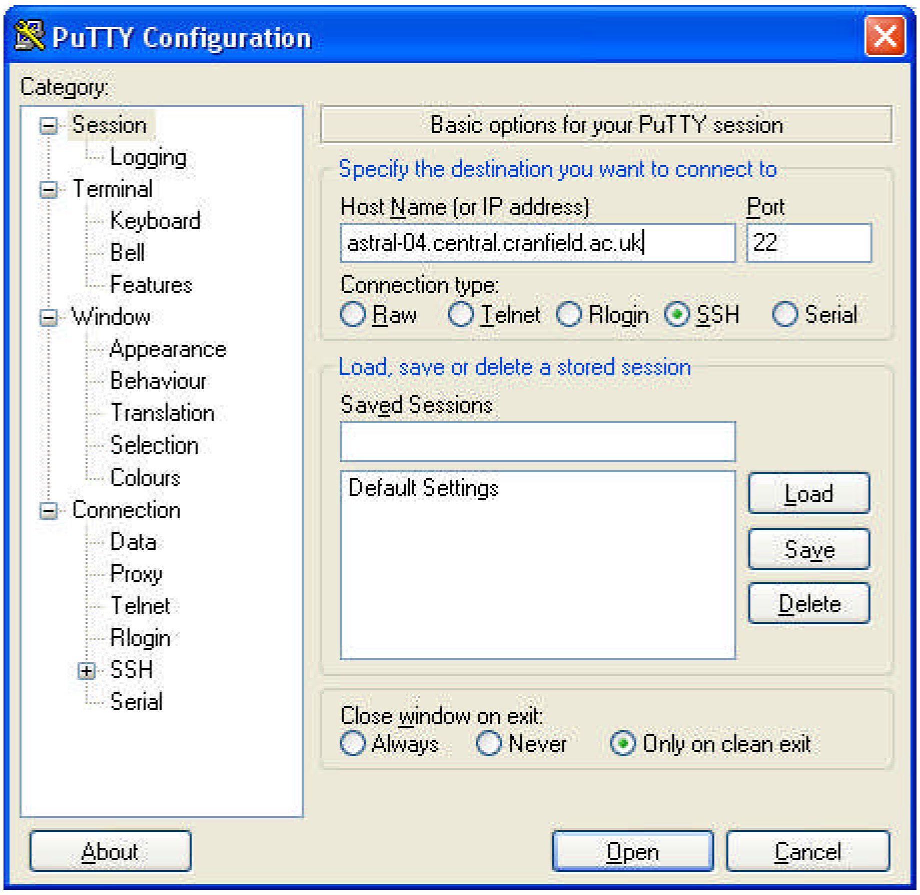

Host Name (or IP address): astral-04.central.cranfield.ac.uk
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Saved Sessions: Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

Figure 24 Putty configuration interface

Next you just need to input your user name and password. Then you will connect to the host you specified. But sometimes the specified host could be unavailable, and you will be redirected to another host. If you trust this host, choose to connect to, if not just exit and try to connect to a host you trust.

There are a few commands we need to use for the experiments.

- module list

  This command line is used to display a list of all the modules loaded.
- module load mpi

  This command line is used to load MPI module.
- module load ddt

  This command line is used to load DDT module which is a graphical debugger that is designed for debugging parallel applications.
- ddt

  This command line is used to run DDT module to debug parallel applications.
- mpicc –o world world.c –g

  This command line is used to compile and link the "world.c" file. "mpicc" is used to compile and link programs written in C++. "-o" stands for linking the output and making an executable. Because the program is written in C++, it must be built with "-g" flag. If math library is used in the program "-lm" flag should be added.
- mpirun –np 2 ./world …

  This command line is used to run the specified program "world" on two processors. "-np" stands for "number of processors". "…" stands for the arguments needed by the program. The program runs without any argument, then "…" should be empty.
- gedit

  This is an official text editor, which is used to view and edit the source code.

If you need to use DDT to debug the parallel applications, there are a few steps to configure DDT. The instructions are downloaded from Astral intranet site.

5.2 Design of Experiments

After the settings, we connect to Astral via Windows now and compile and run parallel programs written in MPI. The aim of these experiments is to check the performance of the parallel program written in MPI. So let's first analyze what factors could have influences on the performance of the parallel programs. And then according to the factor analysis design the experiments.

5.2.1 Factor Analysis

What factors have influences on the performance of parallel programs? Here we have a list of these factors, some are general factors for all of the parallel programs, and some are specific factors which are applied to this project. First, let's have a look at the general factors.

- Load Balancing

  Speed-up ratio is generally limited by the slowest node. So ensuring that each processor is given the same amount of work is very important for achieving high performance.

- Communication

  Basically speaking, the communication and calculation cannot be overlapped, and then any time spent on communication directly degrades the speed-up ratio. We also get the conclusion that "The effect of communication on speedup is reduced, in relative terms, as the grain size increase." The parallel algorithm designer should make the suitable size of sub tasks, so that we keep all the processors busy and reduce the total time of communications.

- Amdahl's Law

  Amdahl's Law is named after Gene Amdahl, who is the designer of IBM 360 and 370. Amdahl's Law is expressed as:[26]

$$\bar{S}_p \le \frac{1}{r_s + \frac{r_p}{n}}$$

where $r_s + r_p = 1$ and r_s represents the ratio of the sequential portion in one program, r_p represent the parallel portion in one program, and n represents the number of processors. The Amdahl's Law points out that the speed-up ratio is limited by the portion of sequential operations in the program.

These are all the general factors which have impact on the parallel performance. The list below lists the specific factors in this project.

- Size of Problems

  This factor is also defined as a general factor. But in this project, the size of the problems is also referred as the size of the original dataset needed to be solved. In the Data Generation section, we have list the tests datasets used in this project. It is easy to get the conclusion that larger problems will need more times when the other factors keep the same. For example, for the same parallel algorithm using same processors, the time for solving problem of 50×50 grid must be less than the time for solving problem of 100×100 grid.

- Spatial Pattern of Dataset

  Spatial pattern is an uncertain factor. We don't know whether it will impact the parallel performance. From the code we have implemented, how the attribute values are distributing across the whole dataset has nothing to do with the computation time. The program just runs the operations on the data no matter what is the value of the data. In order to exclude this factor, we will have a serial of experiments on the dataset with same grid size but different spatial patterns first of all. If the time used for these experiments are the same, then it means that the spatial pattern has nothing to do with the parallel performance.

Here we list five factors, how to deal with these factors will be discussed next.

The load balancing requires that the sub-domains are of comparable sizes. From the discussion in Parallel Algorithm section, we have the conclusion that the sub domains defined by the program are of comparable sizes, so this should not be the factor limits the developed parallel code.

In Parallel Algorithm section, we also analyze the communication factor. The communications needed in the project are broadcast and gather, which are collective communications. We use the domain decomposition method to partition the spatial weight matrix into strips in rows, and the calculation for each strip is independent with other strips. This means that there is no communications between any processors during the calculation. The only communications arise at the beginning and end of the process. At the beginning, the root processor needs to broadcast the original dataset to all of the processors. At the end, the root processor needs to gather results from all the processors in the rank order.

Talking about the Amdahl's Law, it is not very important for this project; we don't take it into considerations.

For the size of problems, we design to do the experiments on datasets with different grid sizes. Then conclusions would be given after the analyzing of the results.

For the spatial pattern factor, as we mentioned before, how the attribute values are distributing across the whole dataset has nothing to do with the computation time. But we will have a serial of experiments on the dataset with same grid size but different spatial patterns to improve our thoughts.

In summary, there will be two serials of experiments: experiments with different grid size; and experiments with same grid size but different spatial patterns.

5.2.2 Experimental Methods

In this section, we will introduce how to do both of the experiments mention before. The experiments with same grid size but different spatial patterns will be done first. This is to exclude the spatial pattern factor for the next experiments.

- Experiments with Different Spatial Patterns

  In order to do these experiments, we need to prepare the tests datasets with same grid size but different spatial patterns. We set four groups of experiments. Each group has the same datasets. These datasets have same grid size but different spatial pattern. Each group will run with different number of processors. We need to record the execution time for each of the experiments, then compare and analyze the results. Table 6 lists the experiments with different spatial patterns.

Table 6 Experiments with different spatial patterns

Group ID	Grid Size	Dataset Name	Number of Processor
1	50×50	data_25_0	1
		data_25_1	
		data_25_2	
2	50×50	data_25_0	2
		data_25_1	
		data_25_2	
3	50×50	data_25_0	4
		data_25_1	
		data_25_2	
4	50×50	data_25_0	8
		data_25_1	
		data_25_2	

- Experiments with Different Grid Size

  In order to do these experiments, we need to prepare the tests datasets with different grid sizes. As the experiments about the spatial pattern have already been done, for each grid size we only need one dataset. Here, we use the random datasets. There are three groups of experiments. Each group will run with different numbers of processors with same dataset. Just like the experiments with different spatial pattern, we also need to record the execution time for each of the experiments, then compare and analyze the result.

Table 7 Experiments with different grid size

Group ID	Grid Size	Dataset Name	Number of Processor
5	50×50	data_25_0	1
			2
			4
			8
6	100×50	data_50_0	1
			2
			4
			8
7	100×100	data_100_0	1
			2
			4
			8

We have mentioned that command line “mpirun –np 2 ./world …” is used to run the specified program using specific number of processors. The number followed the “-np” flag is the number of processors.

Then we start to do these experiments and record the execution time for each of the experiment. Next section is the results discussion. We will see a serial of tables and graphs which show the execution time for each experiment and the speed-up ratio for different number of processors.

5.3 Results Discussion

As we have defined two kinds of experiments, in the discussion part, we will discuss the results of both the experiments separately. First of all, we will have a discussion about the results of the experiments with different spatial patterns, then the results of the experiments with different grid sizes.

5.3.1 Experiments with Different Spatial Patterns

In the Design of Experiments section, we list the experiments we need to do for discovering the relationship between the spatial pattern and the performance of parallel algorithm. There are four groups of experiments. Each group did the experiments with same grid size using same number of processors. We have four tables to list the results of each group.

Table 8 shows the results of Group 1. The experiments in this group using only one processor to calculate Moran’s I for datasets with same grid size but different spatial patterns. The execution time for each of the experiment is about 157-158 seconds.

Table 8 Results of experiments in Group 1

Group ID	Grid Size	Dataset Name	Number of Processor	Time (s)
1	50×50	data_25_0	1	157.45
		data_25_1	1	158.13
		data_25_2	1	157.91

Table 9 shows the results of Group 2. The experiments in this group were similar to that in Group 1 but using two processors. It is clear that the computing time for each of the experiment in this Group is almost the same, just like the results in Group 1.

Table 9 Results of experiments in Group 2

Group ID	Grid Size	Dataset Name	Number of Processor	Time (s)
2	50×50	data_25_0	2	78.58
		data_25_1	2	78.99
		data_25_2	2	78.70

Table 10 shows the results of Group 3, which using four processors to calculate Moran's I for the same datasets used in Group 1 and Group 2. We also get the same conclusion as Group 1 and Group 2.

Table 10 Results of experiments in Group 3

Group ID	Grid Size	Dataset Name	Number of Processor	Time (s)
3	50×50	data_25_0	4	39.32
		data_25_1	4	39.33
		data_25_2	4	39.48

Table 11 shows the results of Group 4. Experiments in this group make use of eight processors to calculate Moran's I. And the execution time for the experiment in this group is similar to others'.

Table 11 Results of experiments in Group 4

Group ID	Grid Size	Dataset Name	Number of Processor	Time (s)
4	50×50	data_25_0	8	20.84
		data_25_1	8	20.12
		data_25_2	8	19.89

First, just have a look at the tables separately. From Table 8 to Table 11, we see that the experiments with same grid size using same number of processors have almost the same execution time.

Second, combining these tables and a conclusion is given. The conclusion is that no matter how many processors are used for the calculation. The execution time for solving problems with same size should be almost the same. The spatial pattern has nothing to do with the performance of the program.

By the way, as the parallel program only implements one kind of spatial weight matrix which is Distance-based Spatial Weight Matrix. This is because execution time for a Distance-based Spatial Weight Matrix is longer than other spatial weight matrices. By using Distance-based Spatial Weight Matrix, it is easier to represent the speedup-ratio of a parallel program. So the conclusions are suitable for Distance-based Spatial Weight Matrix only. If you want to know conclusions for other spatial weight matrices, the parallel program should also implement other spatial weight matrices, and the same experiments should be done again.

5.3.2 Experiments with Different Problem Sizes

For the experiments with different problem sizes (or different grid size) we want to know how the grid size impacts on the performance. Also we are very interested with the speed-up ratio on n processors. In the Design of Experiments section, we have list the experiments we need to do. There are three groups of experiments. Each group does the experiments with same grid dataset but using different number of processors. For each group we list a table to show the parallel speed-up ratio.

Table 12 Results of experiments in Group 5

Group ID	Grid Size	Dataset Name	Number of Processor	Time (s)	Speed-up Ratio
5	50×50	data_25_0	1	157.62	1.00
			2	79.26	1.99
			4	40.02	3.94
			8	21.45	7.35

The experiments of Group 5 are using the same dataset, but with different number of processors. For a dataset which contains 2500 grid cells, the serial time of the calculating the Moran's I is about three minutes. As the size of the problem is fixed, the decline in the speed-up ratio is expressed as the number of processors increase can decrease in computation per processor. So the execution time for each processor decreases. Because the communication does not increase much, the speed-up ratios are very close to the numbers of processors.

Table 13 Results of experiments in Group 6

Group ID	Grid Size	Dataset Name	Number of Processor	Time (s)	Speed-up Ratio
6	100×50	data_50_0	1	1255.06	1.00
			2	628.84	1.99
			4	316.57	3.96
			8	183.13	6.85

The experiments in Group 6 are using the same dataset with 5000 grid cells. Just have a look at the results of Group 6, and we get almost the same conclusion as we got in Group 5. But here we need to notice that when we are using eight processors, the efficiency decreases. In Group 5 we also have this situation, but not such marked. In the parallel program, we ask for printing the execution time for each processor as well as the total execution time. From the screen, we find out some of the processors take more time than others, this is a problem caused by load balancing. And this problem causes a decrease in speed-up ratio as well as efficiency.

Table 14 Results of experiments in Group 7

Group ID	Grid Size	Dataset Name	Number of Processor	Time (s)	Speed-up Ratio
7	100×100	data_100_0	1	9893.13	1.00
			2	5001.13	1.98
			4	2523.13	3.92
			8	1392.51	7.10

The experiments in Group 7 are using the same dataset with 10000 grid cells. The execution time for one processor is close to three hours. In this case, using parallel computing really saves time. During the experiments, sometimes an error will occur which said the program is stopped by the host for some reason. So please pay attention when you are doing these experiments.

Finally, we want to have a look at the speed-up ratio comparison of different problem size. See Figure 25, no matter what is the problem size, the speedup-ratio of the parallel program is almost the same when using same number of processers.

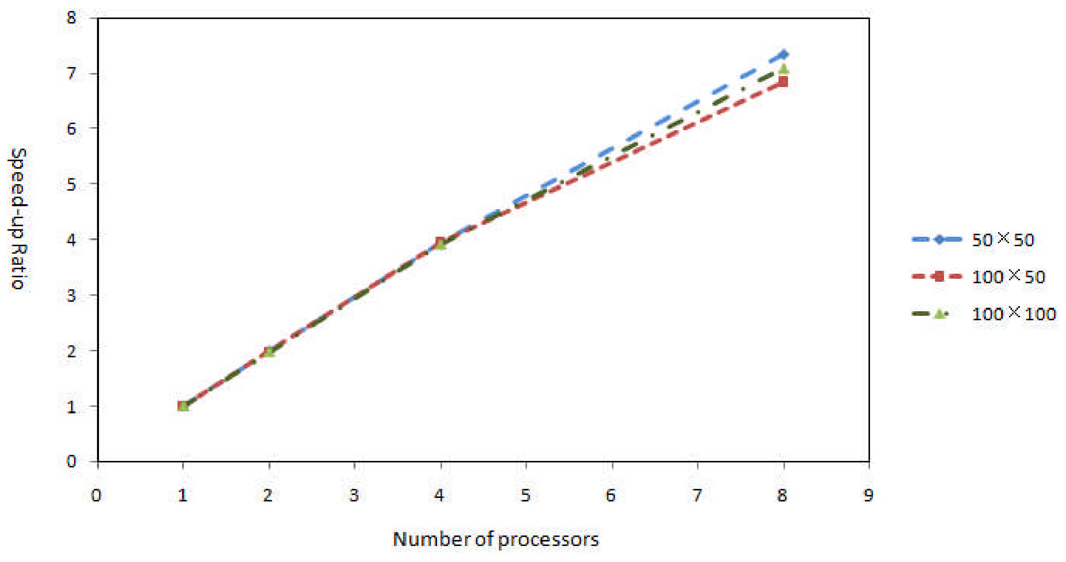


Figure 25 Speed-up comparison of different size

Figure 26 compares the execution time of the experiments using same number of processors but with different grid size. From Figure 24, we know that when the number of the grid cells increases, the execution time is about N^3 times, and N is the proportion of different grid size. For example, the execution time of the experiment using 50×50 dataset on two processors is about 79 seconds, and the execution time of the experiments using 100×50 dataset on two processors is about 628 seconds. The proportion equals to 7.9 which is close to 2^3, while the second dataset contains two times of grid cells contained in the first dataset.

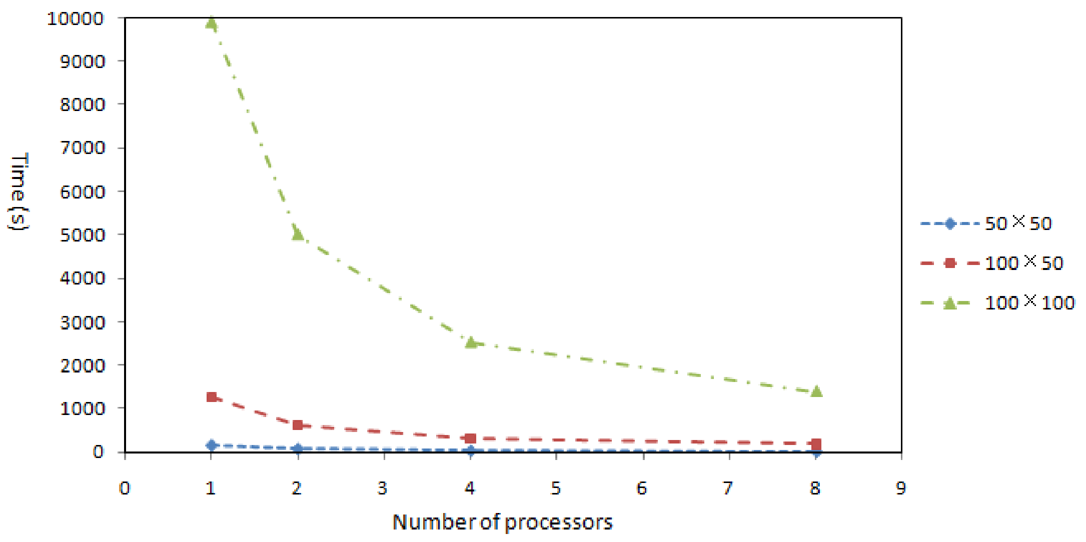


Figure 26 Time comparison of different size

5.4 Summary

In this chapter, we describe the test environment and how to do the experiments in details. The factors limits the performance have been analyzed. According to the experiments, we get such conclusions, which instruct us to improve the parallel algorithm in future.

- Spatial pattern has nothing to do with the parallel performance. This is the conclusion for the Distance-based Spatial Weight Matrix only. Because the elements in Distance-based Spatial Weight Matrix are non-zero except the diagonal elements. Basically speaking, no matter how the attribute values are distributed across the dataset, the number of the operations needed for the calculations keeps the same. But spatial pattern may have influence on the performance if we use other spatial weight matrices, Fixed Distance Band Weight Matrix for example. For the Fixed Distance Band Weight Matrix, if a cluster exists in the dataset, the larger cluster is, more non-zero elements exist in the spatial weight matrix and more operations needed to be done, as a result more execution time is spent.

- The problem size is a limitation on performance. As the number of the grid cells increases, the execution time is about N^3 times, N is the proportion of different grid size. But the limitation is just about the execution time, the efficiencies of the experiments with same number of processors are almost the same. But when there are more processors, instability of the performance will occur, just like the experiments using eight processors.
- Load balancing would be another limitation on performance. The domain decomposition used in the project is a kind of static load distribution. We try to make sure that the sub-domains are of comparable sizes. This aim has almost been achieved, because execution time used by each processor is similar.

6. Tests in Grid Environment

Just like the introduction of grid computing in Literature Review chapter, there are many definitions of grid computing. However, one of the most used toolkits for creating and managing a grid environment is the Globus Toolkit. The Globus Toolkit is an open source software toolkit used for building grids. The latest stable release is Globus Toolkit 5.0.2. As the aim of this project is to use grid computing technologies to solve spatial analysis problems, so we build a simple grid environment with the help of Sun VirtualBox and Globus Toolkit.

The first part of this chapter describes how to build a simple grid environment for testing. This environment is built with Sun VirtualBox and Globus Toolkit (the version is Globus Toolkit 4.2.1 release).

The second part of this chapter discusses how to design the experiments and how to use the Globus Toolkit to enable programs to run in the grid environment built at first. In this project, we focus on how to submit the jobs in grid environment, so GRAM is the main component used in the project, for specific, GRAM4 is used. A few components are used by GRAM4, such as: Reliable File Transfer (RFT); GridFTP; Delegation; local job scheduler; and sudo. So we must have GridFTP and RFT working, if we want to make use of GRAM4.

The third part is about the results discussion. And finally, a summary of this chapter is given.

6.1 Tests Environment

How to build a simulated grid environment? This is the problem needed to be solved in this section. Actually a grid environment can be built with computers connected with network. In this simulated gird environment three virtual machines are contained at least. So a virtualization application is needed. We also need an application to build and management the grid environment, Globus Toolkit 4 is chosen to meet the requirements. This section presents the installation and configuration of grid environment step by step. The grid environment is shown in Figure 27:

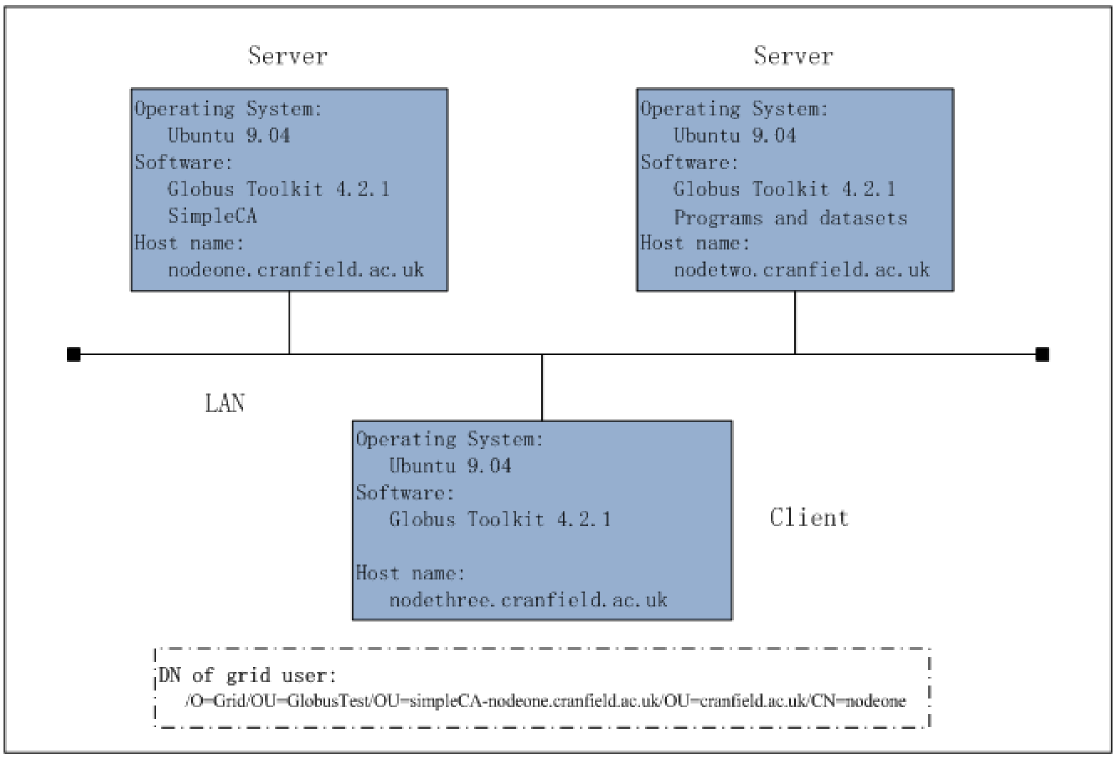


Figure 27 Hardware and software environment of each machine

6.1.1 Setup Virtual Machines

Sun VirtualBox a virtualization application which is installed on Windows, Mac, Linux or Solaris operating systems. With VirtualBox we setup three virtual machines under a host operating system.

Table below summarizes the names (contain long name and short name) of the machines to be used in the grid, and their IP addresses.

Table 15 Host names and IP addressing

Host Name (long)	Host Name (short)	IP Address
nodeone.cranfield.ac.uk	nodeone	192.168.27.181
nodetwo.cranfield.ac.uk	nodetwo	192.168.27.156
nodethree.cranfield.ac.uk	nodethree	192.168.27.172

As we don't have any name servers, so it is necessary to list such information in the /etc/hosts hosts file with the following lines, adjusting the IP addresses to match those assigned for your grid machines:

```
192.168.27.181  nodeone.cranfield.ac.uk    nodeone
192.168.27.156  nodetwo.cranfield.ac.uk    nodetwo
192.168.27.172  nodethree.cranfield.ac.uk  nodethree
```

There are three columns, first is the IP address assigned to the machine, second is the host name (long version), third is the host name (short version). Make sure the long name is before the short name; if not Globus will make mistakes when finding a fully qualified names for IP addresses.

After setting up these virtual machines, reboot them and verity machine connectivity, using the ping command to ping each of the other machines by name or IP address.

All these virtual machines use the same operating system: ubuntu-9.04-desktop-i386. There is an UltraISO file named "ubuntu-9.04-desktop-i386.iso" used for the installation of operating system.

6.1.2 Setup Grid Environment

From the last part, we know there are three virtual machines act as grid machines, how to setup and configure the grid environment is the problem we need to solve in this section. First, we will shortly introduce the software installed on each machine. And then some details about the installation. The instructions about the installation are listed in appendix part.

- *nodeone*

  We need to install Globus Toolkit and Globus Simple Certificate Authority (referred as SimpleCA). This machine acts as a Grid server as well as an administrator of certificates. This node is in charge of managing and generating certificates for the users and servers in this grid environment.

- *nodetwo*

  This is another grid node. Globus Toolkit is installed in this machine. To be specific, GRAM should be setup and configured in this machine. The CA's certificate should be installed, especially requesting and signing certificates for servers. The steps of the installation will be introduced later.

- *nodethree*

  This machine acts as the client in this grid environment. For a user who will use the grid, a few commands must be executed. There commands are used to request and sign user certificates. The installation will be introduced step by step.

Before the installation, let’s make one thing clear. The users’ names are different on different machines, but they share the same grid user ID, which is Distinguished Name (referred as DN):

```
/O=Grid/OU=GlobusTest/OU=simpleCA-nodeone.cranfield.ac.uk/OU=cranfield.ac.uk/CN=nodeone
```

DN is used by the CA in the grid environment, which is defined by CCITT 1988 recommendation x.509. In a real grid environment, the distinguished name should be globally unique.

The Globus Toolkit installations are the same for these machines, so we assume that the installations of Globus Toolkit are finished. As our operating system is Ubuntu, “gt4.2.1-x86_deb_4.0-installer.tar.gz” is chosen for the installation. Now we are going to setup elements of the grid environment.

The work we are going to do require authentication and authorization, and certificates are used for this purpose. You must request certificates for your servers’ processes from your CA. Next, set up your user certificate and validate the connection between the user and applications and servers. The users need their key pair to be authenticated by the grid servers with a certificate (usercert.pem) and private key (userkey.pem).

- Setup Security

  On machine *nodeone*, we need to install SimpleCA. A hash number is generated and used as a part of the file name, just like the file name below:

  ```
  globus_simple_ca_72fbae51_setup-0.20.tar.gz
  ```

  And then complete setup of GSI. Now request and sign a host certificate. Users also need to request user certificate, and the SimpleCA owner (on machine *nodeone* under globus) must sign the user certificate. After that the SimpleCA owner should create a “grid-mapfile”, and add this user to the grid file.

On machine *nodetwo*, we also need to setup the security. The file which named "globus_simple_ca_72fbae51_setup-0.20.tar.gz" is generated on machine *nodeone*. This is the file containing the public CA key and other information needed to participate in this grid. So copy this file and install it. As machine *nodetwo* is another server in this grid, which is going to run services on it, so it will need its own host certificates (host certificates on machine *nodetwo* are owned by root) and grid-mapfile as well. But the user certificates (user certificates on machine *nodetwo* are owned by nodetwo) can be reused. Just copy the user certificates from machine *nodeone* to machine *nodetwo*. The user certificates are listed below:

```
-rw-r--r-- 1 nodetwo nodetwo 2755 2010-07-11 23:57 usercert.pem
-rw-r--r-- 1 nodetwo nodetwo 1411 2010-07-11 23:58 usercert_request.pem
-r-------- 1 nodetwo nodetwo  963 2010-07-12 00:01 userkey.pem
```

On machine *nodethree*, file "globus_simple_ca_72fbae51_setup-0.20.tar.gz" is needed. Because it contains enough information needed to participate in grid. Machine *nodethree* acts as a client, so we are not going to run services on it.

- Setup Services

GRAM4 is the main component used in the project. A few components are used by GRAM4, such as: Reliable File Transfer (RFT); GridFTP; Delegation; local job scheduler; and sudo. So we must setup GridFTP, configure RFT, and setup sudo to make GRAM4 working.

The sudo configuration file is in /etc/ directory. The "sudoers" file is given:

```
Runas_Alias GLOBUSUSERS = ALL, !root;
globus ALL=(GLOBUSUSERS) NOPASSWD: /usr/local/globus/libexec/globus-gridmap-and-execute -g /etc/grid-security/grid-mapfile /usr/local/globus/libexec/globus-job-manager-script.pl *
globus ALL=(GLOBUSUSERS) NOPASSWD: /usr/local/globus/libexec/globus-gridmap-and-execute -g /etc/grid-security/grid-mapfile /usr/local/globus/libexec/globus-gram-local-proxy-tool *
```

Create the gsiftp configuration file in the /etc/xined.d/ directory. After the configuration, restart xinetd. The gsiftp configuration file is given:

```
service gsiftp
{
        instances = 100
        socket_type = stream
        wait = no
        user = root
        env +=GLOBUS_LOCATION=/usr/local/globus
        env += LD_LIBRARY_PATH=/usr/local/globus/lib
        server = /usr/local/globus/sbin/globus-gridftp-server
        server_args = -i
        log_on_success += DURATION
        disable = no
}
```

Generally speaking, if you need to setup a server, the steps are list:

- Installation of Globus Toolkit
- Setup of server certificates
- Setup of user certificates
- Server setup

But if this server is also the owner of SimpleCA, then SimpleCA must be installed. Here machine *nodeone* acts as this role.

If we are going to setup a client machine, then we just need to do:

- Installation of Globus Toolkit
- Setup of user certificates

6.1.3 Check the installation

First, check the installations on each machine. Then for a server machine, using following commands to see if the GridFTP are listening on a port.

```
netstat –an | grep 2811
```

If it is listening, then a line below will display:

```
tcp        0      0 0.0.0.0:2811           0.0.0.0:*               LISTEN
```

Form the client machine (machine *nodethree*) logged on as the user nodethree, do the following:

- Setup the environment

  ```
  . $GLOBUS_LOCATION/etc/globus-user-env.sh
  ```

- Refresh the proxy certificate

  ```
  grid-proxy-init
  ```

- Send a job to the server machine and return the result. Here we need to use such commands on machine *nodethree*, because it is the client machine, and machine *nodeone* and machine *nodetwo* are server machines.

  ```
  globusrun-ws -submit -F nodeone.cranfield.ac.uk -c /bin/true
  globusrun-ws -submit -F nodetwo.cranfield.ac.uk -c /bin/true
  ```

6.2 Design of Experiments

After the setting up, we have built the grid environment which is used to do experiments. The aim of these experiments is to prove that the spatial analysis problems are solved in grid environment. As GRAM is the main component used in the project, first introduction of GRAM will be given, and the experimental methods will be described in details.

6.2.1 Basic Conceptions

GRAM is a Globus project that produces technologies which enable users to locate, submit, monitor and cancel remote jobs on Grid-based computer resources. GRAM is meant to address a range of jobs where arbitrary programs, reliable operation, stateful monitoring, credential management, and file staging are important. Job management with GRAM makes use of multiple types of service: job management services (used to represent, monitor, and control the overall job life); file transfer services (supports staging of files into and out of compute resources); and credential management services (used to control the delegation of rights).

Talking about GRAM, we need to know about job submission conception in GRAM. GRAM services provide secure job submission to job schedulers. First of all, a valid proxy is required for job submission, which is generated using *grid-proxy-init* tool. The job submission is considered as a process of submitting a job description to the GRAM services. And GRAM provides command-line tools which run on user side to submit jobs. The official job submission client for GRAM4 is *globusrun-ws* (a C program for submitting and managing jobs to a Grid host).

Jobs submitted to GRAM services have a lifetime. The clients have full control over the lifetime of the jobs. If a client does not specify a lifetime then the job will not be removed until it is fully processed, regardless how long the job takes.

As we mentioned before, jobs submitted to GRAM4 services must be described in GRAM4 job description language (which is RSL – Resource Specification Language). The Globus provides Job Description Schema documentation which defines all elements used in a job description XML file. Figure 28 shows a sample of such an XML file.

```
<job>
    <executable>my_echo</executable>
    <directory>${GLOBUS_USER_HOME}</directory>
    <argument>Hello</argument>
    <argument>World!</argument>
    <stdout>${GLOBUS_USER_HOME}/stdout</stdout>
    <stderr>${GLOBUS_USER_HOME}/stderr</stderr>
    <fileStageIn>
        <transfer>
            <sourceUrl>gsiftp://job.submitting.host:2811/bin/echo</sourceUrl>
            <destinationUrl>file:///${GLOBUS_USER_HOME}/my_echo</destinationUrl>
        </transfer>
    </fileStageIn>
    <fileStageOut>
        <transfer>
            <sourceUrl>file:///${GLOBUS_USER_HOME}/stdout</sourceUrl>
            <destinationUrl>gsiftp://job.submitting.host:2811/tmp/stdout</destinationUrl>
        </transfer>
    </fileStageOut>
    <fileCleanUp>
        <deletion>
            <file>file:///${GLOBUS_USER_HOME}/my_echo</file>
        </deletion>
    </fileCleanUp>
</job>
```

Figure 28 An example job description with file staging

6.2.2 Experimental Methods

According to last section, we know that GRAM4 supports job execution with coordinated file staging on a remote machine in the grid. And the executable programs and files used are stored distributed across the grid environment.

In this grid environment, there are only three machines, two servers and one client. In the real world, the executable programs and the datasets are stored on a remote machine sometimes, so we suppose that the executable programs and datasets are stored on the server machine which is machine *nodetwo* in this project, and the executable program used for calculating Moran's I is going to run on the server machines (machine *nodeone*). Meanwhile when the computations are finished, related files must be deleted.

As we mentioned before, the jobs submitted to GRAM4 services must be described in GRAM4 job description language with an XML file. In this test, the executable programs and files are stored on a server machine (machine *nodetwo*), but run on server machines (machine *nodeone*), and the results will be returned to client machine (machine *nodethree*), so the job must contain file staging processes. In order to do file staging, some specific elements (such as *fileStageIn* and *fileStageOut*) must be contained in the job description file. The file transfer directives contain a source URL and destination URL. URLs point to remote files as well as local files, so that we access to any file allowed in the grid environment. Because the copied files and executable programs related to computations must be deleted, another element (which is *fileCleanUp* element) must be contained in the job description file.

The process of the experiment is shown in Figure 29. From this graph, we see that the datasets and programs are stored in machine *nodetwo*, machine *nodethree* is a client machine, and machine *nodeone* is the machine where the program is running on. First, the client (machine *nodethree*) with valid certificate runs the GRAM client programs and submits a job to a GRAM server (machine *nodeone*). Then GRAM server asks for datasets and programs. The server (machine *nodetwo*) with these data will transfer them to the GRAM server (machine *nodeone*). After the computing, the results will be returned to client, and related files on GRAM server will be deleted.

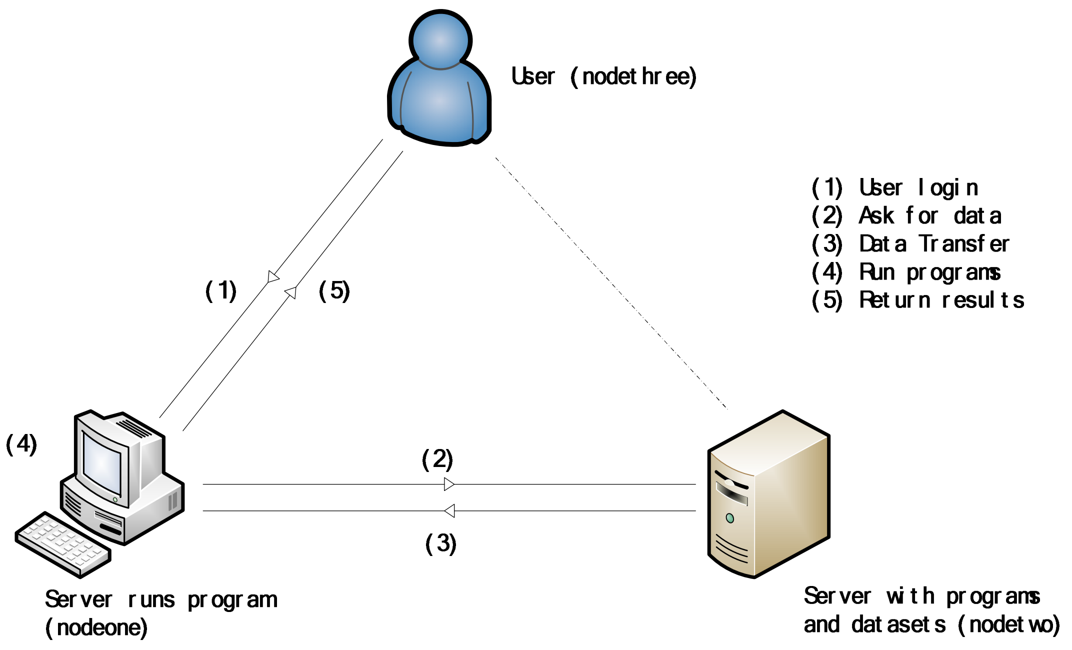


Figure 29 Workflow of experiments in Grid environment

Table 16 lists the experiments in this Grid environment. As the program just run on only one machine, so we just use the tests datasets to check whether the results are correct, and how the work is going on. The serial program is used in this test. Execution time is needed to record for the latter discussion.

Table 16 Experiments in Grid environment

Group ID	Grid Size	Dataset Name
8	50×50	data_25_0
		data_25_1
		data_25_2
9	100×50	data_50_0
		data_50_1
		data_50_2
10	100×100	data_100_0
		data_100_1
		data_100_2

6.3 Results Discussion

Table 17 lists the results of the experiments. The computing time for each of the experiment is recorded. Here Distance-based Spatial Weight Matrix is used in the experiments.

Table 17 Results of experiments in Group 8-10

Group ID	Grid Size	Dataset Name	Time (s)
8	50×50	data_25_0	146.52
		data_25_1	147.13
		data_25_2	150.74
9	100×50	data_50_0	1214.94
		data_50_1	1215.47
		data_50_2	1195.26
10	100×100	data_100_0	9960.19
		data_100_1	9875.76
		data_100_2	10052.01

The reason of using Distance-based Spatial Weight Matrix is the elements in it are non-zero except the diagonal elements, which means that efficient loop structure cannot decrease the computing time. Also, we use the same spatial weight matrix in tests in Astral environment, it is better to use the same one in tests.

From the table, we see that the time used for each of the dataset with same grid size is close to others'. This is a same conclusion as the experiments in Astral environment. So, in a grid environment, the spatial patterns have no influence on the performance.

Secondly, comparing the computing time with different size, the computing time in Group 9 is almost 8 times as that in Group 8. Similarly, the computing time in Group 10 is also about 8 times as that in Group 9. It means, when the size of the dataset increases, the execution time will increase. And the portion is about N^3 times, and N is the proportion of dataset size.

Comparing the results with that in Astral environment, we find an interesting thing, which is the execution time in grid environment for 50×50 dataset is shorter than that in Astral environment. According to the analysis, it should because of the data transferring. In the simulated grid environment, we use bridged adapter. And when we use “ping” command to test the network connection, there is no latency. I think this is the reason. But we need to realize, in a real grid environment, latency must exist, and the situation about the network will be more complex. That is why test in real grid environment is needed in future.

6.4 Summary

Though we use Globus Toolkit for this project, it is important to know that there are other software used for building and managing a grid environment. In this chapter, we build a simulated grid environment, and use this environment to simulate a distributed GIS system. The datasets and programs are stored on a remote machine. The client just sends the job description to the GRAM server (which does not have datasets and programs), and waits for the results. The GRAM server will ask for data transferring and then execute the spatial analysis. The experiments show that data transferring and program executing are finished correctly, and the results returned to the user are correct.

7. Conclusions and Further Work

7.1 Final Conclusions

In this project, we try to use grid computing technology and parallel program technology in spatial analysis. The usage is illustrated based on a case study of Moran's I which is a measure of spatial autocorrelation in spatial analysis. Algorithms for calculating Moran's I statistic values have been developed. In order to investigate the performance of the algorithms, we have two kinds of tests: one is test in Astral environment; the other is test in Grid environment. Through the tests, we get such conclusions:

- Efficient loop structures are very necessary for the calculating of Moran's I. This is because the calculations are related to the spatial weight matrix, and most of the spatial weight matrices are sparse matrices, which means a lot of zeros contained in the matrices. Efficient loop structures benefit decreasing computing time.
- When developing parallel algorithm, decomposition of the spatial weight matrix benefit the parallelism. As calculations in each part are independent.
- Grid technology benefits distributed GIS. Spatial analysis programs and spatial data are stored separately. Meanwhile spatial analysis runs on other machines. And the client acts as a thin client.
- Finally, the most important conclusion is that grid computing and parallel computing technologies can be used in solving computing intensive and memory intensive problems. The calculation of Moran's I statistic is such a problem, so we take it as a case study.

We also get conclusions about the Moran's I algorithms. First of all, the performance of the algorithms with different kinds of spatial weight matrices is different. For a sparse spatial weight matrix, the performance will be better. Secondly, the performance of the parallel algorithms won't be influenced by the spatial patterns.

7.2 Further work

In this study, we focus on developing algorithms for calculating Moran's I statistic values, and the tests in Astral and Grid environment, so we ignore a few details which should be considered in a real GIS system, such as the implementation of other spatial analysis methods and so on. Here we list the future work need to do.

- Supports for other kinds of spatial data

  In this project, raster data model is used. And the datasets used in this project are stored in a .txt file. The reason to use this kind of datasets is to simplify the operations on dataset files. But in a real GIS system or application, it must support for different kinds of spatial data. There are a serial of standard spatial data formats, such as GML (Geography Markup Language); Shapefile; Binary; AutoCAD DXF; JPEG 2000 and so on.
- Implementation of other spatial analysis methods

  As we have mentioned, spatial analysis is a complex conception with different methods. A real spatial analysis application should contain the most widely used spatial analysis methods. Moran's I is one of them. Others like Geary's C (another measures of spatial autocorrelation); Kriging Interpolation (a famous spatial interpolation methods) should also be implemented.
- Tests in a real grid environment

  In Chapter Tests in Grid Environment, we build a simulated grid environment, and run tests in this simulated grid environment. But the real grid environment will be more complex which contains thousands of distributed computers and other devices connected by network belong to different virtual organizations. And only applications in a real grid environment are used by wide users.
- Building a grid portal for spatial analysis

  Portals are important components in grid computing that provide an open and standards-based user interface to grid middleware.[27] A grid portal provides an easy-to-use manner for the grid users, especially when the users are not familiar with grid. So building a grid portal for spatial analysis will benefit the using of the system.

References

[1] Foster, I. and Kesselman, C. (1999), *The grid: blueprint for a new computing infrastructure*, Morgan Kaufmann, San Francisco.

[2] Foster, I., Kesselman, C. and Tuecke, S. (2001), "The anatomy of the grid: Enabling scalable virtual organizations", *International Journal of High Performance Computing Applications*, vol. 15, no. 3, pp. 200-222.

[3] DeFanti, T. A., Foster, I., Papka, M. E., Stevens, R. and Kuhfuss, T. (1996), "Overview of the I-way: Wide-area visual supercomputing", *International Journal of High Performance Computing Applications*, vol. 10, no. 2-3, pp. 123-131.

[4] Joseph, J., Ernest, M. and Fellenstein, C. (2004), "Evolution of grid computing architecture and grid adoption models", *IBM Systems Journal*, vol. 43, no. 4, pp. 624-645.

[5] Flynn, M. J. (1972), "SOME COMPUTER ORGANIZATIONS AND THEIR EFFECTIVENESS.", *IEEE Transactions on Computers*, vol. C-21, no. 9, pp. 948-960.

[6] Vivek Sarkar (1989), *Partitioning and Scheduling Parallel Programs for Multiprocessors*, MIT Press, Cambridge, MA, USA.

[7] Dueker, K. J. and Kjerne, D. (1989), "Multipurpose cadastre: terms and definitions", *Agenda for the 90's.Technical papers 1989 ASPRS/ACSM annual convention, Baltimore.Vol.5*, , pp. 94-103.

[8] Sun, Q., Chi, T., Wang, X. and Zhong, D. (2005), "Design of Middleware based Grid GIS", Vol. 2, pp. 854.

[9] Shaowen Wang, Armstrong, M. P., Jun Ni and Yan Liu (2005), "GISolve: a grid-based problem solving environment for computationally intensive geographic information analysis", *Challenges of Large Applications in Distributed Environments, 2005. CLADE 2005. Proceedings*, pp. 3.

[10] Shekhar, S., Ravada, S., Kumar, V., Chubb, D. and Turner, G. (1996), "Parallelizing a GIS on a shared address space architecture", *Computer,* vol. 29, no. 12, pp. 42-48.

[11] Hunsaker C, Ehlschlaeger CR, Davis FW and Goodchild MF (1996), "Estimating spatial uncertainty as a function of scale: implications for landscape ecology", *Third International Conference/Workshop on Integrating GIS and Environmental Modeling,* January 21-25, 1996, Santa Fe, New Mexico, USA, .

[12] Magillo, P. and E. Puppo (1998), "Algorithms for Parallel Terrain Modelling and Visualisation", in Richard Healey, Steve Dowers, Bruce Gittings, et al (eds.) *Parallel Processing Algorithms for GIS,* Taylor and Francis, London, pp. 351-386.

[13] Armstrong, M. P., Cowles, M. K. and Wang, S. (2005), "Using a computational grid for geographic information analysis: A reconnaissance", *Professional Geographer,* vol. 57, no. 3, pp. 365-375.

[14] Moran, P. A. (1950), "Notes on continuous stochastic phenomena.", *Biometrika,* vol. 37, no. 1-2, pp. 17-23.

[15] Anselin, L. (1995), "Local indicators of spatial association - LISA", *Geographical Analysis,* vol. 27, pp. 93-115.

[16] Anselin, L. (2005), "Spatial Statistical Modeling in a GIS Environment", in David J. Maguire, Michael Batty and Michael F. Goodchild (eds.) *GIS, Spatial Analysis, and Modeling,* ESRI Press, California, USA, pp. 93-111.

[17] Anselin, L. (1988), *Spatial Econometrics: Methods and Models,* Kluwer, Boston, USA.

[18] Getis, A. (2009), "Spatial weights matrices", *Geographical Analysis,* vol. 41, no. 4, pp. 404-410.

[19] Berry, B. and Marble, D. (eds.) (1968), *Spatial Analysis,* Prentice Hall, Englewood Cliffs.

[20] Walford, N. (1995), *Geographical data analysis,* Wiley, England.

[21] Jian Lian, Huili Gong, Xiaojuan Li, Yonghua Sun, Wenhui Zhao and Lin Zhu (2009), "The analysis of economic spatial characteristics of Beijing-Tianjin-

Hebei metropolitan region based on GIS", *Geoinformatics, 2009 17th International Conference on,* pp. 1.

[22] Anselin, L. (1998), "Exploratory spatial data analysis in ageocomputational environment", in P.Longley, S.BRooks, B.Macmillan, et al (eds.) *Geo Computation: Aprimer,* Wiley, NewYork, USA, pp. 77-94.

[23] Cliff, A. and J. K. Ord (1981), *Spatial Processes: Models and Applications,* Pion Ltd, London.

[24] Tobler W. (1970), "A computer movie simulating urban growth in the Detroit region", *Economic Geography,* vol. 46, no. 2, pp. 234-240.

[25] Mukaddes, A. M. M. and Uragami, A. (2008), "Parallel performance of domain decomposition method on distributed computing environment", *Computer and Information Technology, 2008. ICCIT 2008. 11th International Conference on,* pp. 617.

[26] Amdahl, G. M. (1967), "Validity of the single processor approach to achieving large scale computing capabilities", *AFIPS '67 (Spring): Proceedings of the April 18-20, 1967, spring joint computer conference,* Atlantic City, New Jersey, ACM, New York, NY, USA, pp. 483.

[27] Silva, V. (2005), *Grid computing for developers,* Charles River Media, London.

Appendices

The appendix is about how to setup a grid environment with Sun VirtualBox and Globus Toolkit. As we have three machines in the grid environment, we will introduce how to setup and configure these three machines separately.

1. Setup the first machine

1.1 Required Software

- Globus Toolkit installer

  The gt4.2.1-x86_deb_4.0-installer.tar.gz is chosen for the project. It contains GPT. First download the installer to our own computer, preparing for the next installation.

- Java Development Kit

  Download the installer from the sun java website, and save the installer in the location "/usr/local/". The installer named jdk-6u20-linux-i586.bin is chosen.

```
root@nodeone:/usr/local# chmod a+x jdk-6u20-linux-i586.bin
root@nodeone:/usr/local# ./jdk-6u20-linux-i586.bin
```

- Ant 1.6.2+

  Download the installer in the location "/usr/local/". As root, install ant:

```
root@nodeone:/usr/local# tar zxvf apache-ant-1.8.1-bin.tar.gz
```

- C compiler

  As root of the system, type command line below:

```
root@nodeone:~# apt-get install g++
```

- Openssl 0.9.7

  As ubuntu doesn't contain the development part, so libssl-dev is chosen instead. As root of the system, typing command line below:

```
root@nodeone:~# apt-get install libssl-dev
```

There are a few software which already been installed, for example GUN tar, GUN make, GUN sed, and Zlib. These are contained in the Ubuntu System. And GPT does not need to be installed as it is contained.

- Set up ANT_HOME and JAVA_HOME

  Set up a few variables, such as ANT_HOME and JAVA_HOME. And these variables are saved in file ***/etc/profile***.

```
export JAVA_HOME=/usr/local/jdk1.6.0_20
export PATH=$JAVA_HOME/bin:$PATH
export ANT_HOME=/usr/local/apache-ant-1.8.1
export PATH=$ANT_HOME/bin:$PATH
```

1.2 Installing GT 4.2.1

- Create a new user

  Now we are ready to install GT 4.2. First create a new user named "globus", and create a directory for the installation of GT 4.2. The "globus" user should be the owner of this directory and has corresponding permissions.

```
root@nodeone:/usr/local# adduser globus
Adding user `globus' ...
Adding new group `globus' (1001) ...
Adding new user `globus' (1001) with group `globus' ...
Creating home directory `/home/globus' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@nodeone:/usr/local# mkdir /usr/local/globus
root@nodeone:/usr/local# chown globus:globus /usr/local/globus/
```

- Build the toolkit

  In order to install toolkit correctly, a new variable GLOBUS_LOCATION should also be set up. And then run "make" and "make install" command to build the toolkit.

```
root@nodeone:/usr/local# vi /etc/profile
export GLOBUS_LOCATION=/usr/local/globus
export PATH=$GLOBUS_LOCATION/bin:$PATH

root@nodeone:/usr/local# su globus
globus@nodeone:/usr/local$ source /etc/profile
globus@nodeone:/usr/local$ cd /usr/local/gt4.2.1-x86_deb_4.0-installer/
globus@nodeone:/usr/local/gt4.2.1-x86_deb_4.0-installer$ ./configure --
prefix=$GLOBUS_LOCATION
checking for javac... /usr/local/jdk1.6.0_20/bin/javac
checking for ant... /usr/local/apache-ant-1.8.1/bin/ant
configure: creating ./config.status
config.status: creating Makefile
globus@nodeone:/usr/local/gt4.2.1-x86_deb_4.0-installer$ make
globus@nodeone:/usr/local/gt4.2.1-x86_deb_4.0-installer$ make install
```

- Check the installation

```
globus@nodeone:/usr/local/gt4.2.1-x86_deb_4.0-installer$ source
/usr/local/globus/etc/globus-user-env.sh
globus@nodeone:/usr/local/gt4.2.1-x86_deb_4.0-installer$ globus-version
4.2.1
```

1.3 Insatlling SimpleCA

- Change host name and IP address

```
root@nodeone:/usr/local $ vi /etc/hostname
nodeone.cranfield.ac.uk
root@nodeone:/usr/local $ vi /etc/hosts
192.168.27.181  nodeone.cranfield.ac.uk    nodeone
192.168.27.156  nodetwo.cranfield.ac.uk    nodetwo
192.168.27.172  nodethree.cranfield.ac.uk  nodethree
```

- Set up SimpleCA

```
globus@nodeone:/usr/local/gt4.2.1-x86_deb_4.0-installer$ source
/usr/local/globus/etc/globus-user-env.sh
globus@nodeone:/usr/local/gt4.2.1-x86_deb_4.0-installer$
/usr/local/globus/setup/globus/setup-simple-ca


------------------------------------------------------------------------
   C e r t i f i c a t e   A u t h o r i t y   S e t u p

This script will setup a Certificate Authority for signing Globus
users certificates.  It will also generate a simple CA package
that can be distributed to the users of the CA.

The CA information about the certificates it distributes will
be kept in:

/home/globus/.globus/simpleCA/

The unique subject name for this CA is:

cn=Globus Simple CA, ou=simpleCA-nodeone.cranfield.ac.uk, ou=GlobusTest, o=Grid
*****************************************************************************
setup-ssl-utils: Complete
```

Now, SimpleCA has already been installed, but there is still one more step. As there is a note, which is “To complete setup of the GSI software you need to run the following script as root to configure your security configuration directory”.

- Set up GSI

```
root@nodeone:/usr/local/gt4.2.1-x86_deb_4.0-installer# source /etc/profile
root@nodeone:/usr/local/gt4.2.1-x86_deb_4.0-installer#
/usr/local/globus/setup/globus_simple_ca_72fbae51_setup/setup-gsi -default
```

- Request for host certificate

```
root@nodeone:~# grid-cert-request -host nodeone.cranfield.ac.uk
Generating a 1024 bit RSA private key
................++++++
...................++++++
writing new private key to '/etc/grid-security/hostkey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Level 0 Organization [Grid]:Level 0 Organizational Unit [GlobusTest]:Level 1
Organizational Unit [simpleCA-nodeone.cranfield.ac.uk]:Name (e.g., John M. Smith) []:

A private host key and a certificate request has been generated
with the subject:

/O=Grid/OU=GlobusTest/OU=simpleCA-
nodeone.cranfield.ac.uk/CN=host/nodeone.cranfield.ac.uk
```

Then the "root" user has a series of certificates.

- Sign the certificate for "globus" user

```
globus@nodeone:/usr/local/globus/bin$ ./grid-ca-sign -in hostcert_request.pem -out
hostcert.pem

To sign the request
please enter the password for the CA key:
```

```
The new signed certificate is at: home/globus/.globus/simpleCA//newcerts/01.pem
root@nodeone:/usr/local/globus/bin# cp /usr/local/globus/bin/hostcert.pem /etc/grid-
security
root@nodeone:/usr/local/globus/bin# cd /etc/grid-security/
root@nodeone:/etc/grid-security# cp hostkey.pem containerkey.pem
root@nodeone:/etc/grid-security# cp hostcert.pem containercert.pem
root@nodeone:/etc/grid-security# chown globus:globus containerkey.pem
containercert.pem
```

- Sign the certificate for a normal user

```
nodeone@nodeone:/etc/grid-security$ source /usr/local/globus/etc/globus-user-env.sh
nodeone@nodeone:/etc/grid-security$ grid-cert-request

globus@nodeone:/etc/grid-security$ source /usr/local/globus/etc/globus-user-env.sh
globus@nodeone:/etc/grid-security$ cd /home/nodeone/.globus/
globus@nodeone:/home/nodeone/.globus$ grid-ca-sign -in usercert_request.pem -out
signed.pem

The new signed certificate is at: /home/globus/.globus/simpleCA//newcerts/02.pem

root@nodeone:/home/globus/.globus/simpleCA/newcerts# cp 02.pem
/home/nodeone/.globus/usercert.pem
root@nodeone:/home/globus/.globus/simpleCA/newcerts# chown nodeone:nodeone
/home/nodeone/.globus/usercert.pem
```

- Check whether the certificate is correct.

```
nodeone@nodeone:/home/globus/.globus/simpleCA/newcerts$ source
/usr/local/globus/etc/globus-user-env.sh
nodeone@nodeone:/home/globus/.globus/simpleCA/newcerts$ grid-proxy-init -debug -
verify
```

```
User Cert File: /home/nodeone/.globus/usercert.pem
User Key File: /home/nodeone/.globus/userkey.pem

Trusted CA Cert Dir: /etc/grid-security/certificates

Output File: /tmp/x509up_u1000
Your identity: /O=Grid/OU=GlobusTest/OU=simpleCA-
nodeone.cranfield.ac.uk/OU=cranfield.ac.uk/CN=nodeone
Enter GRID pass phrase for this identity:
Creating proxy ...++++++++++++
......++++++++++++
 Done
Proxy Verify OK
Your proxy is valid until: Mon Jul 12 07:40:34 2010
```

- Create a grid-mapfile

```
root@nodeone:/etc/grid-security# grid-mapfile-add-entry -dn
/O=Grid/OU=GlobusTest/OU=simpleCA-
nodeone.cranfield.ac.uk/OU=cranfield.ac.uk/CN=nodeone -ln nodeone

New entry:
"/O=Grid/OU=GlobusTest/OU=simpleCA-
nodeone.cranfield.ac.uk/OU=cranfield.ac.uk/CN=nodeone" nodeone
(1) entry added
root@nodeone:/etc/grid-security# cat /etc/grid-security/grid-mapfile
"/O=Grid/OU=GlobusTest/OU=simpleCA-
nodeone.cranfield.ac.uk/OU=cranfield.ac.uk/CN=nodeone" nodeone
```

1.4 Setup first machine: GridFTP

- Create the gsiftp configuration file in the /etc/xined.d/ directory and configure it.

```
root@nodeone:~# cat /etc/xinetd.d/gridftp
service gsiftp
{
instances = 100
socket_type = stream
wait = no
user = root
env +=GLOBUS_LOCATION=/usr/local/globus
env += LD_LIBRARY_PATH=/usr/local/globus/lib
server = /usr/local/globus/sbin/globus-gridftp-server
server_args = -i
log_on_success += DURATION
disable = no
}

root@nodeone:~# tail /etc/services
tfido           60177/tcp                       # fidonet EMSI over telnet
fido            60179/tcp                       # fidonet EMSI over TCP

# Local services
myproxy-server 7512/tcp #Myproxy server
gsiftp      2811/tcp
root@nodeone:~# /etc/init.d/xinetd reload
 * Reloading internet superserver configuration xinetd                  [ OK ]
root@nodeone:~# netstat -an | grep 2811
tcp       0      0 0.0.0.0:2811          0.0.0.0:*             LISTEN
```

1.5 Setup first machine: GRAM4

- Configure sudo

  We have to setup sudo so the globus user can start jobs as a different user. The configuration is list:

```
root@nodeone:/home/nodeone# cat /etc/sudoers

Runas_Alias GLOBUSUSERS = ALL, !root;
globus ALL=(GLOBUSUSERS) NOPASSWD: /usr/local/globus/libexec/globus-gridmap-
and-execute -g /etc/grid-security/grid-mapfile /usr/local/globus/libexec/globus-job-
manager-script.pl *
globus ALL=(GLOBUSUSERS) NOPASSWD: /usr/local/globus/libexec/globus-gridmap-
and-execute -g /etc/grid-security/grid-mapfile /usr/local/globus/libexec/globus-gram-
local-proxy-tool *
```

- Check whether the GRAM has been set up correctly.

```
nodeone@nodeone:~$ globusrun-ws -submit -c /bin/true
Submitting job...Done.
Job ID: uuid:9577be52-8d2d-11df-8c34-080027cac2fd
Termination time: 07/11/3010 20:47 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
nodeone@nodeone:~$ echo $?
0
```

2. Set up second machine

2.1 Install software for the second machine

- Create a new user

  First of all, create a new user named “globus”, a directory for the installation of GT 4.2. The “globus” user should be the owner of this directory and has corresponding permissions.

- Change host name and IP address

```
root@nodetwo:/usr/local $ vi /etc/hostname
nodetwo.cranfield.ac.uk
```

```
root@nodetwo:/usr/local $ vi /etc/hosts
192.168.27.181  nodeone.cranfield.ac.uk    nodeone
192.168.27.156  nodetwo.cranfield.ac.uk    nodetwo
192.168.27.172  nodethree.cranfield.ac.uk  nodethree
```

- Install software

  In the second machine, we also need to install Java, Ant, and Globus Toolkit. The software should be installed just like the first machine. Finally, just check whether the software is installed correct.

```
globus@nodetwo:/usr/local/gt4.2.1-x86_deb_4.0-installer$ ./configure --
prefix=$GLOBUS_LOCATION
checking for javac... /usr/local/jdk1.6.0_20/bin/javac
checking for ant... /usr/local/apache-ant-1.8.1/bin/ant
configure: creating ./config.status
config.status: creating Makefile
globus@nodetwo:/usr/local/gt4.2.1-x86_deb_4.0-installer$ make
globus@nodetwo:/usr/local/gt4.2.1-x86_deb_4.0-installer$ make install

globus@nodetwo:/usr/local/gt4.2.1-x86_deb_4.0-installer$ globus-version
4.2.1
```

2.2 Setup second machine: Security

- Install SimpleCA for the second machine

  Copy the “simple_ca_72fbae51_setup-0.20.tar.gz” file from nodeone to nodetwo. And save the file in the directory: “/home/globus”, then install it.

```
globus@nodetwo:~$ gpt-build -force globus_simple_ca_72fbae51_setup-0.20.tar.gz
globus@nodetwo:~$ su root
Password:
root@nodetwo:/home/globus#
/usr/local/globus/setup/globus_simple_ca_72fbae51_setup/setup-gsi -default
setup-gsi: Configuring GSI security
Making /etc/grid-security...
```

```
mkdir /etc/grid-security
Making trusted certs directory: /etc/grid-security/certificates/
mkdir /etc/grid-security/certificates/
Installing /etc/grid-security/certificates//grid-security.conf.72fbae51...
Running grid-security-config...
Installing Globus CA certificate into trusted CA certificate directory...
Installing Globus CA signing policy into trusted CA certificate directory...
setup-gsi: Complete
```

- Request for the host certificate

```
root@nodetwo:/home/globus# source /usr/local/globus/etc/globus-user-env.sh
root@nodetwo:/home/globus# grid-cert-request -host nodetwo.cranfield.ac.uk
```

- Sign the host certificate for the second machine

  Send the host certificate to nodeone, and let nodeone sign the host certificate for nodetwo. And then send the signed certificate to nodetwo.

```
globus@nodeone:/mnt/share$ source /usr/local/globus/etc/globus-user-env.sh
globus@nodeone:/mnt/share$ grid-ca-sign -in hostcert_request.pem -out hostcert.pem
```

- Copy the user certificates from the first machine to the second machine

  These certificates should be stored in /home/nodetwo/.globus.

```
root@nodetwo:/etc/grid-security# ls -l *.pem
-rw-r--r-- 1 globus globus 2747 2010-07-11 23:53 containercert.pem
-r-------- 1 globus globus  887 2010-07-11 23:52 containerkey.pem
-rw-r--r-- 1 root   root   2747 2010-07-11 23:47 hostcert.pem
-rw-r--r-- 1 root   root   1402 2010-07-11 23:44 hostcert_request.pem
-r-------- 1 root   root    887 2010-07-11 23:44 hostkey.pem

root@nodetwo:/home/nodetwo/.globus# ls -l
-rw-r--r-- 1 nodetwo nodetwo 2755 2010-07-11 23:57 usercert.pem
-rw-r--r-- 1 nodetwo nodetwo 1411 2010-07-11 23:58 usercert_request.pem
-r-------- 1 nodetwo nodetwo  963 2010-07-12 00:01 userkey.pem
```

- Verify the certificate for the second machine

```
nodetwo@nodetwo:~/.globus$ grid-proxy-init -verify –debug
```

- Create the grid-mapfile for the second machine

```
root@nodetwo:/etc/grid-security# source /usr/local/globus/etc/globus-user-env.sh
root@nodetwo:/etc/grid-security# grid-mapfile-add-entry -dn
/O=Grid/OU=GlobusTest/OU=simpleCA-
nodeone.cranfield.ac.uk/OU=cranfield.ac.uk/CN=nodeone -ln nodetwo
Modifying /etc/grid-security/grid-mapfile ...
/etc/grid-security/grid-mapfile does not exist... Attempting to create /etc/grid-
security/grid-mapfile
New entry:
"/O=Grid/OU=GlobusTest/OU=simpleCA-
nodeone.cranfield.ac.uk/OU=cranfield.ac.uk/CN=nodeone" nodetwo
(1) entry added
root@nodetwo:/etc/grid-security# cat /etc/grid-security/grid-mapfile
"/O=Grid/OU=GlobusTest/OU=simpleCA-
nodeone.cranfield.ac.uk/OU=cranfield.ac.uk/CN=nodeone" nodetwo
```

2.3 Setup second machine: GridFTP

- Copy the “/etc/xinetd.d/gridftp” and “/etc/services” from the first machine to the second machine, and then reload xinetd.

```
root@nodetwo:/mnt/share# /etc/init.d/xinetd reload
 * Reloading internet superserver configuration xinetd                    [ OK ]
```

2.4 Setup second machine: GRAM4

- Copy the “/etc/sudoers” file from the first machine to the second machine. And then test the GRAM:

```
nodetwo@nodetwo:~$ globusrun-ws -submit -F nodeone.cranfield.ac.uk -c /bin/true

Submitting job...Done.

Job ID: uuid:3b74c05a-919a-11df-8720-080027a08d66

Termination time: 07/17/3010 11:55 GMT

Current job state: Active

Current job state: CleanUp

Current job state: Done

Destroying job...Done.
```

3. Set up third machine

The third machine acts as a client, so the installation of the third machine is much more easily. We just need to install Globus Toolkit4, and the user certificates for the third machine.

3.1 Install GT 4.2.1

The installation of GT 4.2.1 is just the same as we have done on the first and second machine. First of all create a new user.

- Create a new user

  Create a new user named "globus", and create a directory for the installation of GT 4.2.
- Change host name and IP address

```
root@nodethree:/usr/local $ vi /etc/hostname

nodethree.cranfield.ac.uk

root@nodethree:/usr/local $ vi /etc/hosts

192.168.27.181  nodeone.cranfield.ac.uk    nodeone

192.168.27.156  nodetwo.cranfield.ac.uk    nodetwo

192.168.27.172  nodethree.cranfield.ac.uk  nodethree
```

- Install software

  In the third machine, we also need to install Java, Ant, and Globus Toolkit. And the installation is list below:

```
globus@ nodethree:/usr/local/gt4.2.1-x86_deb_4.0-installer$ ./configure --
prefix=$GLOBUS_LOCATION

globus@ nodethree:/usr/local/gt4.2.1-x86_deb_4.0-installer$ make
globus@ nodethree:/usr/local/gt4.2.1-x86_deb_4.0-installer$ make install

globus@ nodethree:/usr/local/gt4.2.1-x86_deb_4.0-installer$ globus-version
4.2.1
```

3.2 Setup third machine: Security

- Install SimpleCA for the second machine

  Copy the "simple_ca_72fbae51_setup-0.20.tar.gz" file from nodeone to nodethree. And save the file in the directory: "/home/globus", then install it.

- Copy the user certificates from the first machine to the third machine. These certificates should be stored in /home/nodetwo/.globus.

- Finally verify the certificate for the third machine

```
nodethree @ nodethree:~/.globus$ grid-proxy-init -verify –debug
```