Computer Science and Application 计算机科学与应用, 2012, 2, 40-46
http://dx.doi.org/10.12677/csa.2012.21008 Published Online March 2012 (http://www.hanspub.org/journal/csa)

Research and Implementation of WebGIS Data Efficient Transmission Technology*

Ye Shen[1,2#], Jing Feng[1], Yueqiang Shu[2], Huiyong Ma[2], Linchun Kang[3]

[1]Institute of Meteorology, PLA University of Science and Technology, Nanjing
[2]Troops 95871 of PLA, Hengyang
[3]Troops 95356 of PLA, Hengyang
Email: #shenye200708@yahoo.com.cn

Received: Dec. 23rd, 2011; revised: Jan. 18th, 2012; accepted: Feb. 1st, 2012

Abstract: There are many factors that affect the transmission performance of WebGIS, the network transmission delay of GIS data increases considerably especially when the network bandwidth is low or the number of service requests is more. For GML map format, this paper proposes a data-driven classified GML compression CGDC (classified GML data compression) which distinguishes the structural data and value data, verifies its effectiveness. For raster data, according to the data size, it verifies how to use tile-based technology to improve data transmission efficiency by experiment.

Keywords: WebGIS; GML; CGDC; Tiles

WebGIS 数据高效传输技术的研究与实现*

沈　晔[1,2#]，冯　径[1]，舒跃强[2]，马辉勇[2]，康林春[3]

[1]解放军理工大学气象学院，南京
[2]95871 部队，衡阳
[3]95356 部队，衡阳
Email: #shenye200708@yahoo.com.cn

收稿日期：2011 年 12 月 23 日；修回日期：2012 年 1 月 18 日；录用日期：2012 年 2 月 1 日

摘　要：影响 WebGIS 传输性能的因素很多，特别是在网络带宽不高或服务请求数较多时，GIS 数据的网络传输延时会急剧增大。针对 GML 地图格式文件，提出了一种数据驱动的分类 GML 压缩方法 CGDC(classified GML data compression)，用以区分结构数据和值数据，并验证其有效性。针对栅格 GIS 数据，根据数据量大小，通过实验验证如何采用瓦片式技术提高数据传输效率。

关键词：WebGIS；GML；CGDC；瓦片技术

1. 引言

WebGIS 深入到了社会的各个领域，然而，其传输性能并不能满足人们的要求，在网络带宽不高或服务请求数较多时，WebGIS 的性能会急剧下降，主要体现在 WebGIS 数据传输的延时很大。减少数据传输量，是在 WebGIS 应用的开发中提高数据传输性能简单实用的技术，但需要优化来提高效率[1]。

GML(Geography Markup Language)是遵循 XML 规范的通用地理数据格式，也是进行数据交换的理想中间数据格式，因此，针对 GML 数据格式进行传输算法的优化研究具有代表意义。根据 GML 文件规范格式，总结出 GML 文件数据结构特征，通过遍历 GML

*资助信息：本文受国家自然科学基金委员会(GrantNo:61070174)和东南大学计算机网络和信息集成教育部重点实验室开放研究基金(grantNo:k939201003)资助。
#通讯作者。

数据，可以分离出结构数据和值数据，并分别进行压缩传输，能有效提高 GML 数据传输效率。

栅格地图是当前流行的 Google 地图、百度地图所采用的数据格式，是 WebGIS 的一个重要发展方向。本文分析了在不同条件下，采用瓦片技术来提高栅格地图数据的传输效率，并得出了有益的性能结果。

本文主要研究了 GML 格式数据的传输优化以及栅格地图的瓦片式传输实现并得出试验结论，文章的其余部分安排如下：第二节介绍了 CGDC 方案思路，详细阐述了 GML 数据的分离、压缩及合并，通过实验进行了验证；第三节采用瓦片技术分析栅格地图文件，并在不同的环境中进行了 WebGIS 性能测试，得出了实验结论。

2. GML 数据传输的优化

2.1. CGDC 方案思路

采用数据压缩技术减少数据传输量，是在网络中优化数据传输常用的方法。基于数据驱动的分类 GML 压缩方法 CGDC(Classified GML Data Compression)，该方法的思路是将 GML 数据分离为结构数据和值数据，并将这两种数据分别输入两个子线程进行压缩处理。结构数据采用 LZ77 压缩算法[2]，值数据采用 Huffman 压缩算法[3]。两个子线程将压缩后的文件在网络中进行传输，在接收端，对压缩后的结构数据和压缩后的值数据，进行结构和值的合并处理，还原成原始 GML 数据[4-8]。如图 1 所示。

2.2. GML 数据解析

目前针对 GML 文件的解析主要分为两种：基于树结构模型和基于事件流模型。

第一种是基于树结构模型，即 DOM 方法，以层次结构组织节点或片段信息，在内存中构建一棵完整的 GML 树。对于大数据量的文件，该方法解析和加载整个文件很慢，且需消耗大量系统资源。而 GML 都是大数据量文件，因此，DOM 解析器不适用于解析 GML 数据。

第二种是基于事件流模型，将文件当作数据流，提供快速、只向前、只读的方式迭代文件中的节点，不需要将整个 GML 文件读入内存中再处理，能有效处理大数据量的 GML 文件。该模型分为：“推模型”和“拉模型”。“推模型”即 SAX 方法，根据解析到的 GML 文件的不同节点，触发一系列的事件，当发现给定的标签时，激活一个回调方法，由应用程序重载该回调函数对节点进行处理。“拉模式”是把感兴趣的节点从读取器中拉出，以程序化方法处理事件，相对应用程序而言，“推模型”是被动接收 GML 元素，“拉模型”是主动接收 GML 元素。

相比“推模型”，“拉模型”的应用更加灵活简单，故本文采用“拉模型”对 GML 文件进行解析。将 GML 文件分离为结构数据 out1 和值数据 out2，同时生成一个用于处理结构数据字典替换的字典表 Dic，简化结构数据。

在对 GML 文件进行解析过程中，结构数据字典替换的步骤为：

1) 如果解析到的标签名或属性名称不存在于字典表中，将其添加进字典表，同时返回一个索引，以后就用这个索引表示该标签或属性名称；

2) 如果解析到的标签名或属性名已经存在于字典表中，直接返回表示该标签或属性名称的索引。

结构数据输出流 out1 由字典表信息 Dic 和插入值数据的标签 Tag 组成，在解析过程中，当解析到唯一开始标签名和属性名称时，向 out1 输入字典表项，当解析到属性值或标签值时，向 out1 输入该插入值数据的标签 Gag，其形成过程如下：

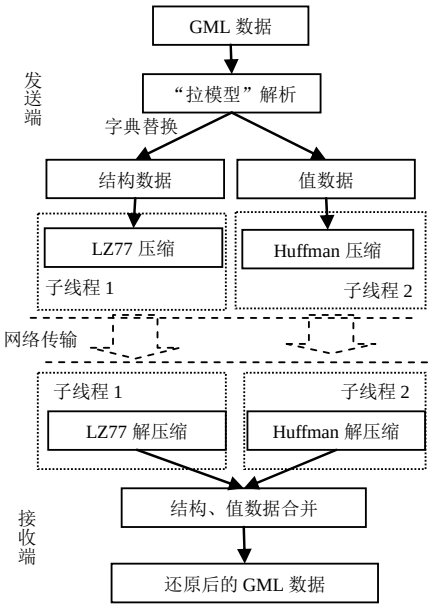

Figure 1. Algorithm processes of CGDC
图 1. CGDC 算法流程

1) 在字典替换过程中，如果标签名或属性名要添加进字典表，那么同时也将其添加 out1 中，即成为了 out1 中的 Dic 信息。

2) 插入值数据的标签 Tag=TE+”,”+TB，其中 TE 为当前插入值与上一个插入值之间需要结束的标签索引集合，如果没有需要结束的标签，则 TE 为空；TB 为当前插入值与上一个插入值之间需要开始的标签或属性索引集合，如果没有需要开始的标签或属性，则 TB 为空。其形成过程为：当解析到的节点类型为结束标签时，将该节点的索引添加到 TE 中，如果节点类型为开始标签或属性时，将该节点的索引添加到 TB 中。其中标签索引用“<N”表示，属性索引用“@N”表示。如 Tag=<4<3，@5<4，表示当前插入值与上一个插入值之间有结束标签 3 和结束标签 4，还有开始标签 4 和属性名 5。注意：第一个插入值标签 Tag 的 TE 为空；最后一个插入值标签的 TB 为空。如果一个插入值标签记 Tag 为“,”，即 TE 和 TB 均为空，表示当前插入值与上一插入值之间没有结构数据，例如一个文本元素的前一个元素为属性值元素，则该文本元素的插入值标签记 Tag 为“,”。

值数据输出流 out2 由属性数据和标签值数据组成。其形成过程比较简单，将“拉模型”解析到的属性数据和标签值数据，依次输入 out2 中。为了便于接收端的合并，应该在属性数据或标签值数据之间插入可用来分隔的特殊字符。

为了更好地理解该解析方法，假设存在如图 2 所示的 GML 片段，对图 2 中 GML 片段的解析结果如表 1 所示。

```
<gml:FeatureCollection>
<gml:featureMember>
<fme:Polygon gml:id="idf2dd19d8">
<fme:OBJECTID>24</fme:OBJECTID>
</fme:Polygon>
</gml:featureMember>
<gml:featureMember>
<fme:Polygon gml:id="id1a477eaa">
<fme:OBJECTID>25</fme:OBJECTID>
</fme:Polygon>
</gml:featureMember>
</gml:FeatureCollection>
```

Figure 2. GML fragment
图 2. GML 片段

Table 1. Resolution of GML
表 1. GML 解析

dictionary	out1	out2
gml:FeatureColltion	gml:FeatureCollection	idf2dd19d8
gml:featureMember	gml:featureMember	24
fme:Polygon	fme:Polygon	id1a477eaa
gml:id	gml:id	25
fme:OBJECTID	,@3<2<1<0	
	fme:OBJECTID	
	,<4	
	<4<2<1,@3<2<1	
	,<4	
	<4<2<1<0,end	

经“拉模型”解析后，生成了结构数据字典表 dictionary、结构数据输出流 out1 和值数据输出流 out2，从表 1 中可以看出，当解析的节点为唯一的标签名或属性名时，将该节点名加入字典表中，同时插入 out1 中，当解析的节点类型为值数据时，将该节点的插入值标签添加到 out1 中，同时将该节点值添加到 out2 中。需要注意的是结构数据字典表 dictionary 只在接收端的内存中起字典替换的作用，不需要传输。对解析后形成的文件分析如下：

1) 字典表中记录了标签名或属性名称与索引的对应关系。

2) 输出流 out1 的可以是字典表中的一个值 Dic，也可以是插入值数据的标记 Tag。它们区分为：前者不以“,”、“@”、“<”开始，后者以这三个字符中的一个开始。

3) 输出流 out2 由大部分高精度数值型字符和少量属性字符值组成。

显然 GML 通过这种方法的解析，能很好的将结构数据和值数据分离开。

2.3. GML 数据压缩

经过字典替换后，GML 源文件结构数据中大量重复的开始标签、结束标签和属性名称，在结构数据输出流 out1 中均只出现一次，大大减少了数据流文件大小，其主要数据为插入值标签，以“<N”或“@N”形式存在，其中大部分是重复出现的；经过字典替换后结构数据中没有了冗余的结构信息，只有一些有规律的标签，为了达到更高的压缩效率，可进行压缩，由于 LZ77 算法的压缩效率高且实现简单，因此结构数据选用 LZ77 算法进行压缩。

值数据流 out2 中，是大部分高精度的数值型字符和少量属性字符组成，这些数据都是无规则的，Huffman 算法是基于统计编码的，适合用于无规则数据的统计压缩，因此值数据采用 Huffman 算法进行压缩。

以上压缩步骤是在两个子线程中进行的，这样充分利用系统资源来提高 GML 传输效率。

2.4. GML 数据合并

根据 2.2 小节设计的结构和值数据的分离存储方法，在接受端，能方便的设计出结构和值数据的合并算法。实际上，在接收端也采用了多线程的方法，对压缩后的结构数据 out1 和值数据 out2 进行解压缩，合并线程是主程序，负责结构数据和值数据的正确合并。

假设两个子线程解压后的输出流为结构数据 out1 和值数据 out2，合并后的输出流为 out3。使用字符串分隔方法，将 out1 和 out2 分别读入两个一维数组 strs1 和 strs2 中，合并算法伪代码如图 3 所示。

在合并算法中，通过整数 k 来控制结构数组和值数组的同步。当结构数组中的元素为插入值标签时，从值数组中提取索引 k 对应的值输出到 out3 中，每次取值完，k 递增 1。

2.5. 实验对比

为了验证算法的高效性，选取了 4 组 GML 数据进行压缩测试实验。为了方便与其他算法的实验结果进行对比，只是在本地进行文件的压缩和解压缩，不考虑图 1 中的网络传输部分。本文选取的对比算法为本文提出的 CGDC 方法、Huffman、LZ77，测试 GML 文件由小到大依次编号 4 组(表 2)。

```
for (int i = 0; i < strs1.Length; i++)//遍历结构数据数组中的元素
{
  string str=",<@";
  if (str.Contains(strs1[1].Substring(0, 1)))//表示该元素为插入值的标签
  {
     string[] strs3 = strs1[1].ToString().Split(',');
     string EndElement = strs3[0].ToString();//结束的标签标记
     string Element = strs3[1].ToString();//开始的标签标记以及属性标记
     GMLAddEndElement(out3, EndElement);//将结束标签添加到out3中
     GMLAddElement(out3, Element);//将开始标签添加到out3中
     if (Element.Substring(0, 1) == "@")
     {//将strs2[k]值作为属性值添加到out3中
          GMLAddAttributeValue(out3, strs2[k].ToString());
     }
     else
     { //将strs2[k]值作为标签文本添加到out3中
           GMLAddValue(out3, strs2[k].ToString());
     }
     k++;
  }
  else//表示该元素为字典表信息
  {
      Dictionary.Add(strs1[1].ToString());//添加到字典表中
  }
}
```

Figure 3. Pseudo code of merging algorithm
图 3. 合并算法伪代码

实验比较结果如图 4、5 所示。

以上实验结果表明，从压缩率来看，本文的 CGDC 方法压缩率最高；从压缩时间来看，CGDC 方法和 LZ77 算法的压缩时间差不多。因此从总的压缩效率来说，CGDC 方法更好。

类似的数据流压缩算法还有采用 SAX 解析 GML 文件[9]，通过本文的分析，SAX 解析方法对大文件的 GML 文件效率不高，当 GML 文件特别大时，很容易出现内存不足而造成系统崩溃。本算法通过“拉模型”解析大文件的 GML 文件，采用只向前、只读方式扫描 GML 文件，效率有明显提高。

Table 2. Test file of GML
表 2. GML 测试样本

文件编号	GML 数据(KB)
1	20.1
2	823
3	6012
4	32,080

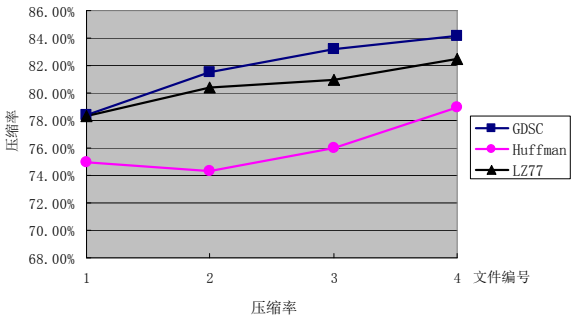

Figure 4. Comparison of compression ratio
图 4. 压缩率对比

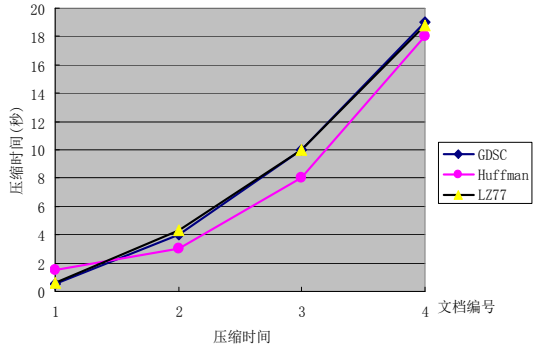

Figure 5. Comparison of compression time
图 5. 压缩时间对比

3. 栅格数据传输的优化

3.1. 设计思路

传统的 WebGIS 请求一次栅格图像，服务器将矢量地图一次性生成栅格图像，这种模式效率低，同时客户端一次性显示一张完整的图像需要大量的时间，这种模式导致地图响应速度慢，浏览器出图延时大，WebGIS 应用性能不高，利用瓦片式 GIS 系统可以很好的解决此问题。

通过开源的 GeoServer 提供的 GWC 服务，在服务器端，事先将 SHP 地图切割成不同比例尺下的瓦片栅格地图，在客户端通过 OpenLayers 进行不同比例尺下一定范围内的瓦片数据请求，如果本地存在该瓦片，则直接从本地读取，否则向服务器获取数据。

3.2. 瓦片式地图服务的创建与发布

目前在 GIS 服务器上建立瓦片缓冲区的 WebGIS 服务平台有 ArcGIS Server、MapGIS IMS FLEX 和 GeoServer。本文采用 GeoServer，它是开源代码的，该特性使得开发者能方便地对服务进行修改、复制以及重发布，其创建发布步骤如下[10]：

1) 安装配置 Java 平台，正确配置环境变量值。

2) 安装、启动 GeoServer，启动 Tomcat。

3) 配置 GeoServer，主要包括服务能力和联系信息配置、WFS、WMS、Data 配置。

4) 配置 GeoWebCache。

5) 在 JS 中发布 GWC 切片服务。

在 GeoWebCache 配置中，可以配置切割地图采用的线程数、瓦片发布模式、采用的地图映射标准、瓦片图片格式、最大最小缩放比例以及定义地图边界，如果不定义地图边界，则切割地图时默认采用 SHP 矢量地图的边界。通过 GeoWebCache 提供的功能，可事先将不同比例尺下的矢量地图数据切割成规定的等大小的栅格图片，当客户端请求一张新地图时，GeoWebCache 将拦截这些调用，返回经过缓存切片处理的地图文件，提高地图显示速度，可减轻 GeoServer 服务器的工作负荷，提供更好的用户体验。

在设置地图显示参数时，应根据实际需要，限定浏览器端只能请求浏览某几个比例尺级别的地图数据，GeoWebCache 可事先将浏览器端可能请求的比例尺下的矢量地图先切割成栅格图片，这样能明显提高客户端的显示效率。

3.3. 系统性能测试

3.3.1. 软硬件环境

系统用于浏览器请求 GIS 服务延时测试的各设备的硬件配置如表 3 所示。由于 WebGIS 服务需要消耗很大的内存，为了减少设备配置低对延时数据采集的影响，所有数据库服务器、应用服务器和客户端均采用较高配置，同时在系统运行过程中，停止各设备中不必要的服务。

局域网环境为：10 MB/S 交换机，双绞线。对用于浏览器请求 GIS 服务延时测试的网络环境进行设置，通过在应用服务器端安装网络管理软件(如 P2P 终结者)，为每台主机的网卡限制最大上行下行速率，来实现不同的局域网网络环境，设置了三种典型的网速：100 KB/s，1024 KB/s 和 10240 KB/s。

系统用于浏览器请求 GIS 服务延时测试的软件环境如表 4 所示。部署在数据库服务器中的 Oracle 数据库存储 GIS 信息的一些附加信息，部署在应用服务器中的 GeoServer 包含各种用于测试的 SHP 地图文件。

3.3.2. 测试数据

系统用于浏览器请求 GIS 服务延时测试的 SHP 数据一共分为 6 组，如表 5 所示。

Table 3. System hardware environment
表 3. 系统硬件环境

设备名称	型号	CPU	RAM	硬盘
数据库服务器	浪潮英信服务器 NP120D	Intel(R) Xeon(R) CPU 5100 @2.2 GHZ	2 GB	500 G SATA
应用服务器	Thinkpad T400	Intel(R) Core(TM)2 Duo CPU P8600 @2.4 GHz	1.98 GB	5400 RPM 250 G
客户端	联想	Pentium(R) 4 CPU 2.6 GHZ	1.00 GB	5400 RPM 160 G

Table 4. System software environment
表 4. 系统软件配置

设备名称	操作系统	支撑软件
数据库服务器	Microsoft Windows Server 2003 Enterprise Edition Service Pack 1	Oracle 10 g
应用服务器	Microsoft Windows XP Professional 2002 Service Pack 3	Tomcat6.0.14、GeoServer2.0.2
客户端	Microsoft Windows XP Professional 2002 Service Pack 3	IE8 浏览器

Table 5. List of system experimental data
表 5. 系统实验数据列表

数据源编号	文件类型	文件大小(KB)
1	点图层	10
2	线图层	200
3	线图层	1123
4	面图层	2607
5	线图层	10,147
6	混合地图	21,780

设置客户端请求比例尺为 1:200 万的地图数据，服务器端事先生成各地图文件的 12 个级别的 png 格式瓦片数据。

3.3.3. 性能分析

在所设定的网络环境中，分别测试了表 5 中的 6 组数据采用瓦片技术和不采用瓦片技术的时延。图 6、图 7、图 8 分别为 100 KB/S、1 MB/S、10 MB/S 网速环境下的时延对比。

从图中的统计信息可知，当 SHP 文件比较小(<1 MB)时，网速对 WebGIS 显示的延迟影响不是很大，而采用 GWC 瓦片缓存技术比不采用 GWC 瓦片缓存技术产生的时延更长，多出的延时是由于采用了 GWC 技术，系统如果没有事先生成瓦片栅格图片，需要先生成栅格图片，其次，SHP 矢量数据本身很小，

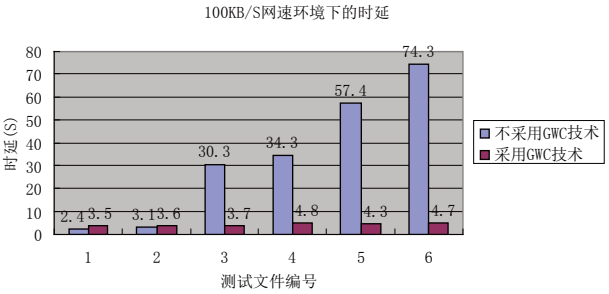

Figure 6. Delay comparison under 100 KB/S
图 6. 100 KB/S 网速环境下的时延对比

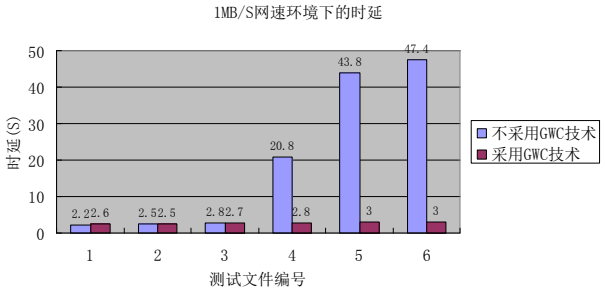

Figure 7. Delay comparison under 1 MB/S
图 7. 1 MB/S 网速环境下的时延对比

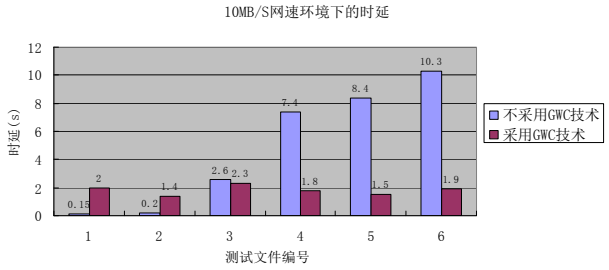

Figure 8. Delay comparison under 10 MB/S
图 8. 10 MB/S 网速环境下的时延对比

生成的瓦片栅格数据总大小反而会比 SHP 矢量数据大，因此就增加了网络传输负担，增大了延时。

当 SHP 文件大于 10 MB 时，在 1 MB/S 和 10 MB/S 的环境中，采用 GWC 技术发布服务，客户端获取响应并显示 GIS 地图，所需要的网络时延要明显低于不采用 GWC 技术。

因此，在通常的 1 M/S 网速环境中，当 SHP 文件比较大时(>10 MB)，GeoServer 服务采用 GWC 瓦片缓存技术会明显提高客户端的显示效果，而在 SHP 文件较小(<10 MB)时，GeoServer 服务没必要采取瓦片缓存技术。

4. 结论

本文讨论了两种格式的地图文件的传输优化方案，针对 GML 格式文件，提出了数据驱动的分类 GML 压缩方法 CGDC，实验表明 CGDC 方法对大数据量的 GML 文件有良好的解析效果，能有效提高 GML 数据在网络中的传输效率，对 GML 文件在 WebGIS 上的推广应用起到了一定的促进作用。针对栅格地图数据，提出采用瓦片技术提高数据传输效率，并在局域网环境中进行了大量验证，得出根据具体的网速环境采取不同的传输方案，以提高数据传输效率。

参考文献 (References)

[1] 杜成龙, 关佶红, 王治. GML 空间数据流压缩算法研究[J]. 计算机工程, 2007, 33(1): 98-100.
[2] C. W. Fraser. An instruction for direct interpretation of LZ77-compressed programs. Software, 2006, 36(4): 397-411.
[3] Y.-N. Wen, G.-H. Lin and S.-J. Chen. Optimal multiple-bit huffman decoding. IEEE Transactions on Circuits and Systems for Video Technology, 2010, 20(5): 621-631.
[4] N. Rishe, O. Wolfson. Schema based XML compression. 2007 International Conference on Enterprise Information Systems and Web Technologies (WIEWT-07), 2007: 1-6.
[5] G. Leighton, T. Muldner and J. Diamond. TREECHOP: A tree-based query-able compressor for XML. Techical Report, Acadia

University, 2005.
[6] M. T. Pankaj, J. P. Myung and W. C. Chin. XPRESS: Aquerable compression for XML data. In: Y. Alon, G. Zachary, Eds., Proceeding of the SIGMOD 2003. San Diego: AC Press, 2003: 122-133.
[7] S. Sakr. XML compression techniques: Asuivey and comparion. Journal of Computer and System Sciences, 2009, 75: 303-322.
[8] Y. Z. Li, I. T. Guan. Spatial data compression techniques for GML. Frontier of Computer Science and Technology, FCST, 2008 Japan-China Joint Workshop on Nagasaki, 2008.
[9] 朱华. GML 空间数据压缩技术研究[D]. 华南师范大学, 2009.
[10] 殷福忠, 孙立民. 基于瓦片金字塔技术的地图发布平台开发研究[J]. 测绘与空间地理信息, 2010, 33(5): 16-17.