西安电子科技大学研究生学位论文

撰写要求 (2015年修订版)

作者姓名 张 三

指导教师姓名、职称 李 四 **选择**

申请学位类别 **选择一项**

西安电子科技大学研究生学位论文

撰写要求 (2015年修订版)

**作者姓名：**张 三

**一级学科：**选择学科

**二级学科：**选择学科

**学位类别：**选择学位

**指导教师姓名、职称：**李 四 职称

**学　　院：** 选择学院

**提交日期：**选取日期

西安电子科技大学

选择类型论文

学　号　 1101110071

密　级　 选择密级

学校代码 10701

分类号 TN82

By

Zhang San

Supervisor: Li Si Professor

February 2015

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of selecting one

in selecting one

Thesis/Dissertation Guide for Postgraduates

of XIDIAN UNIVERSITY

西安电子科技大学

学位论文独创性（或创新性）声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同事对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文若有不实之处，本人承担一切法律责任。

本人签名： 日 期：

西安电子科技大学

关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权属于西安电子科技大学。学校有权保留送交论文的复印件，允许查阅、借阅论文；学校可以公布论文的全部或部分内容，允许采用影印、缩印或其它复制手段保存论文。同时本人保证，结合学位论文研究成果完成的论文、发明专利等成果，署名单位为西安电子科技大学。

保密的学位论文在 年解密后适用本授权书。

本人签名： 导师签名：

日 期： 日 期：

摘要

摘要是学位论文的内容不加注释和评论的简短陈述，简明扼要陈述学位论文的研究目的、内容、方法、成果和结论，重点突出学位论文的创造性成果和观点。摘要包括中文摘要和英文摘要，硕士学位论文中文摘要字数一般为1000字左右，博士学位论文中文摘要字数一般为1500字左右。英文摘要内容与中文摘要内容保持一致，翻译力求简明精准。摘要的正文下方需注明论文的关键词，关键词一般为3～8个，关键词和关键词之间用逗号并空一格。

中文摘要格式要求为：宋体小四、两端对齐、首行缩进2字符，行距为固定值20磅，段落间距为段前0磅，段后0磅。

英文摘要格式要求为：Times New Roman、小四、两端对齐、首行不缩进，行距为固定值20磅，段落间距为段前0磅，段后0磅，段与段之间空一行。

**关 键 词**：XXX， XXX， XXX， XXX， XXX

ABSTRACT

The Abstract is a brief description of a thesis or dissertation without notes or comments. It represents concisely the research purpose, content, method, result and conclusion of the thesis or dissertation with emphasis on its innovative findings and perspectives. The Abstract Part consists of both the Chinese abstract and the English abstract. The Chinese abstract should have the length of approximately 1000 Chinese characters for a master thesis and 1500 for a Ph.D. dissertation. The English abstract should be consistent with the Chinese one in content. The keywords of a thesis or dissertation should be listed below the main body of the abstract, separated by commas and a space. The number of the keywords is typically 3 to 5.

The format of the Chinese Abstract is what follows: Song Ti, Small 4, justified, 2 characters indented in the first line, line spacing at a fixed value of 20 pounds, and paragraph spacing section at 0 pound.

The format of the English Abstract is what follows: Times New Roman, Small 4, justified, not indented in the first line, line spacing at a fixed value of 20 pounds, and paragraph spacing section at 0 pound with a blank line between paragraphs.

**Keywords**: XXX, XXX, XXX, XXX, XXX

插图索引

图序号 插图示例 X

表格索引

表格序号 表格示例 X

符号对照表

符号 符号名称

XXX XXX

XXX XXX

XXX XXX

缩略语对照表

缩略语 英文全称 中文对照

XXX XXX XXX

XXX XXX XXX

XXX XXX XXX

目录

[摘要 I](#_Toc413704332)

[ABSTRACT III](#_Toc413704333)

[插图索引 V](#_Toc413704334)

[表格索引 VII](#_Toc413704335)

[符号对照表 IX](#_Toc413704336)

[缩略语对照表 XI](#_Toc413704337)

[第一章 研究生学位论文撰写的总体要求 1](#_Toc413704338)

[第二章 研究生学位论文撰写的内容要求 3](#_Toc413704339)

[2.1 封面 3](#_Toc413704340)

[2.2 题名页 3](#_Toc413704341)

[2.3 声明 5](#_Toc413704342)

[2.4 摘要 5](#_Toc413704343)

[2.5 插图索引 5](#_Toc413704344)

[2.6 表格索引 6](#_Toc413704345)

[2.7 符号对照表 6](#_Toc413704346)

[2.8 缩略语对照表 6](#_Toc413704347)

[2.9 目录 6](#_Toc413704348)

[2.10 正文 6](#_Toc413704349)

[2.10.1 绪论 7](#_Toc413704350)

[2.10.2 各章节 7](#_Toc413704351)

[2.10.3 结论 8](#_Toc413704352)

[2.11 参考文献 8](#_Toc413704353)

[2.12 致谢 9](#_Toc413704354)

[2.13 作者简介 9](#_Toc413704355)

[2.14 其他 10](#_Toc413704356)

[第三章 研究生学位论文的编辑、打印、装订要求 11](#_Toc413704357)

[3.1 学位论文封面的编辑和打印要求 11](#_Toc413704358)

[3.2 学位论文的版面设置要求 11](#_Toc413704359)

[3.3 学位论文的打印、装订要求 11](#_Toc413704360)

[3.4 其他说明 11](#_Toc413704361)

[第四章 图、表、公式示例 13](#_Toc413704362)

[参考文献 15](#_Toc413704363)

[致谢 19](#_Toc413704364)

[作者简介 21](#_Toc413704365)

# 绪论

## 课题背景

地理信息系统(Geographic Information System, GIS)是处理数字化空间数据的计算机应用系统，是一门以地理学为基础理论依据、以计算机科学为基数支撑、以遥感和GPS为重要数据源的综合性交叉学科，GIS对地理数据进行生成、管理和分析，并在此基础上提供决策分析。目前，随着计算机技术和网络通信技术的迅速发展，GIS为人们提供了一种在全球范围发布和获取信息的新手段。伴随着经济的快速增长与社会各行业的对地理信息需求的日益旺盛，地理信息信息产业正在显现出无限广阔的发展前景与巨大的市场潜力。GIS是一种多学科交叉的产物,它是一个采集、存储、管理、分析、描述和应用整个或部分地球表面（包括大气层在内）与空间和地理分布有关数据的计算机系统。随着全球信息化进程的加快，GIS正从传统的封闭的单机模式向开放和共享的网络化、分布式GIS方向发展，在Internet网络上实现空间信息的发布与共享，已经逐渐成为GIS发展的必然的趋势，网络地理信息系统（Web Geographic Information System，WebGIS）应运而生，并成了目前GIS领域最重要的研究方向之一。

近年来 ,不同部门、不同行业通过遥感、航摄 、GPS、LIDAR 、传统测绘等生产了海量的二维、三维空间地理数据,建立了大量的空间地理信息系统和空间数据库。但是由于空间地理数据的多样性和复杂性,传统的表现方法缺陷日益凸显。各个系统彼此封闭、孤立、缺乏统一标准 ,不同的数据格式、数据结构、存储方法,数据系统和“孤岛数据库”难以实现信息共享和交流,无法实现空问地理信息数据和服务的最大共享。

Web Services 的引入将能改变GIS数据访问的网络互操作，实现真正的网络环境下的空间地理信息共享. 互联网上大量的信息资源多数具有空间分布的特性，利用WebGIS对这些信息进行管理，为用于提供基于空间分布的多种信息服务，提高资源的利用率和共享程度。但是各大厂商都开始不断推出自己的WebGIS，没有一个统一的标准，是很难真正做到在各个平台之间共享各自的数据。 于是OGC（open GIS Consortium开放地理信息系统协会）在XML和web服务基础上制定了一系列用于地理数据的共享和互操作的规范。改规范包含地理信息web服务、GML（Geography Markup Language）以及OWS（OGC Web Services）。地理标识语言GML是OGC推出的基于XML的一种开放式空间数据编码标准。

OGC地理信息服务包括四个主要的地理信息服务:web地图服务（WMS，Web Map Service）、web要素服务（WFS，Web Feature Service）、web覆盖服务（WCS，Web Coverage Service）。切片地图Web服务（WMTS，Web Map Tile Service）。其中WMS属于地图表达服务，用于从网络上的分布式空间数据库中获取地图和要素来形成用户所需的地图。WFS是WMS的进一步深入，WFS返回的是要素级的GML编码。WCS是主要针对的是”Coverage”阿虎局，如卫星影像、数字高程数据等。WMTS标准定义了一些操作，允许用户访问切片地图。

## 国内外发展现状

不同的WebGIS平台具有多样性，其所存储的地图数据结构也有可能不一致，即数据之间存在存储的异构性。针对以上问题，只有选择一种与存储和平台无关的技术才能实现统一、规范和标准的地图数据交换与共享方式。网络地图服务是一种融合分布式网络、计算机图形学、网络服务（WebService）和空间数据库于一体的用于存储、处理和共享地图数据的高新技术，其采用HTTP协议访问与传输数据，与传输两端系统和平台无关，因此，网络地图服务成为WebGIS间地图数据共享的主要途径。在多个WebGIS之间，网络地图数据把地图数据的相关属性与WebService结合起来，根据实际需求，灵活准确地为开发者提供地图数据渲染、分析和可视化表达等一系列接口。开发者基于网络地图服务进行二次开发，辅助决策或实现其他业务功能。目前，网络地图服务已经成为WebGIS中必不可少的基础服务，以其为基础的软件和解决方案正逐步渗透到生产生活中的各个领域，可靠、便捷的网络地图服务正成为WebGIS发展的重要方向之一。

目前，国内外各大公司研制的WebGIS产品中均提供网络地图服务，采用这种服务可以接入Internet提供的网络地图数据，纵使本地没有存储地图数据也可以快速、方便的完成浏览与分析功能，例如国外MapInfo的MapXtreme，ESRI的ArcGIS，Autodesk的MapGuide，Intergraph的GeoMedia WebMap以及开源的GeoGIS等，国内的SuperMap GIS、MapGIS、Citystar等都良好的实现了该类服务。

虽然网络地图服务的发展取得了长足的进步，但是还存在以下不足：

1） 服务中地图发布流程固定，定制性不强；

2） 网络地图服务发布的地图数据源支持格式有限，扩展性不强；

3） 以上产品间的服务中数据交换虽然有一定标准，但仍存在不兼容现象；

4） 某些领域的地图数据存储格式和交换具有特殊标准，以上产品更是无法兼容；

5） 以上大多产品为商业产品，通常需要高昂的服务和维护费用，而一部分开源产品在效率上又达不到要求。

因此，开发一款WebGIS产品，实现定制性与扩展性兼备，并且符合特殊领域地图数据存储和交换标准的网络地图服务系统是必要的。

## 论文的主要工作与章节安排

本文主要阐述了实现高性能分布式的符合OGC标准的GIS服务,设计了由服务展示层、Web服务器层、OGC应用服务器层、数据引擎层 四层架构组成的符合OGC标准的GIS服务。重点阐述了OGC应用服务器层的实现过程，并测试与分析了服务的稳定性和并发性。总结论文所做的工作，论文的章节安排如下：

第一章为绪论部分，简单介绍了课题的研究背景、地理信息系统的发展现状、OGC标准服务规范、分布式OGC服务的四层架构。指出多WebGIS间地图数据交换共享所存在的问题以及相应地解决方案，重点阐述了如何实现高效稳定的分布式OGC应用服务器。

第二章，详细介绍了分布式OGC服务的四层架构，介绍了每层架构的主要功能和相互之间的关联，以及对服务用到的相关技术进行了简单介绍，包括分布式对象中间件Ice、IO模型、内存管理。

第三章，详细介绍了高性能OGC应用服务器的实现，核心阐述了如何实现高效而稳定的服务端。第一,介绍了如何基于分布式对象中间件Ice实现OGC应用服务器层，以及详细对比了几种不同的服务框架模型，列出了各自的优缺点；第二,在服务实现中重点阐述了如何实现高效的内存管理;第三,介绍了服务的预处理技术，从而提高服务的性能和并发性。

第四章，详解介绍了分布式OGC应用服务器层的负载均衡和应用服务器集群的辅助模块的设计，主要包括集群管理监控模块设计和分布式缓存模块设计。

第五章，详细介绍了分布式OGC应用服务层的具体实现，包括OGC应用服务组件的开发、管理监控功能的开发实现以及分布式Memcached缓存模块的实现。对整个分布式OGC服务系统进行功能测试，并对OGC应用服务器层进行更细节的功能和性能测试。

第六章，总结与展望部分，概述系统实现的主要功能，总结了课题研究过程中所做的工作、成果和未来的研究方向。

# 分布式OGC服务框架与功能

## 服务总体框架结构

分布式OGC服务采用服务展现层、Web服务器层、OGC应用服务器层、数据引擎层四层架构组成，采用 Ice 中间件处理应用逻辑，按照 SOA 组建模型的思想，为各种用户提供都中服务OGC标准的地理信息服务。服务的架构图如下2.1:



图2.1 分布式OGC服务系统总体架构

分布式OGC服务采用分层结构，遵循设计模式中的"高内聚、低耦合"的思想。在各种 Web 服务中，这种分层模式的思想都被广泛应用。分层使不同的开发人员关注各自擅长的特定层面，有助于开发优质的系统。分层分离了逻辑，使得系统结构层次明晰，系统变得灵活和易于维护，有助于构建良好的系统。

图2.1中服务展现层是为了更好的演示整个分布式OGC服务，主要用于对外发布数据中心提供的、用户可读的 Web 服务，同时提供服务分类列表、服务关键字检索，并展现服务更新情况、描述服务与示例等。普通用户或者负责二次开发的专业用户可以通过服务展现层获取 OGC 服务系统提供的各类服务的详细功能说明、编程接口说明和功能演示。数据存储系统是用于向整个OGC服务系统提供框架数据支撑。

分布式OGC服务系统的物理框架部署如图2.2，由四个集群组成：Web服务器集群、OGC应用服务器集群、分布式缓存集群、数据集群。服务展现层通过SOA标准从Web服务器集群使用期提供的数据服务，由于分层架构中的每相邻两层都是解耦的，客户端可以开发自己的服务表现层来使用该分布式OGC服务系统。而Web服务器集群是通过分布式对象中间件Ice和OGC应用服务器集群之间进行通信。数据存储系统是由多种不同的数据源组成，目前是下面四种数据源组成：Oracle框架数据源、Sqlite瓦片数据源、本地文件系统数据源、Hadoop云端数据源。在以后可以根据服务的需要动态添加或删除数据源，为了将OGC应用服务器层和这么多不同类型的数据存储系统之间解耦，便开发了数据引擎层。OGC应用服务器集群通过数据引擎层来使用数据存储系统中的数据，而不用关心底层具体的数据源实现。



图2.2 分布式OGC服务框架系统物理部署

## 分布式OGC服务系统各层的功能

根据上一节2.1的服务总体框架的设计，分布式OGC服务自上而下的流程如下： OGC应用服务器端是使用C++语言进行开发，通过数据引擎层实现从数据存储系统获取数据，并使用分布式对象中间件Ice将结果以字节流的形式返回给Web客户端，Web客户端支持各种服务SOAP协议的服务表现层。OGC服务系统支持WFS、WCS、WMS、WMTS、四种OGC服务标准；支持目录服务、元数据服务；支持瓦片数据打包下载服务。

### 服务展现层

在分布式OGC服务框架中，服务展现层是一个基于开源JavaScript库Cesium开发的WebGIS二、三维一体的客户端，是整个分布式OGC服务框架的入口，主要实现的功能是向用户展现一个友好的网络界面，将服务支持的各种数据图层的目录显示在客户端上，用户可以通过客户端操作查看图层元数据、在线浏览地图数据、在线获取要素数据、提交打包下载请求等等。

### Web服务器层

Web服务器端选择了Tomcat作为Web容器，之所以选择Tomcat，是因为Tomcat本身开源、免费、运行时占用系统资源少、有良好扩展性而且对负载均衡的配置有很好的支持，是一款深受广大Web端程序员的偏爱的Web服务器。

Web服务器层主要实现的功能包括服务的注册与发现，用户访问权限的控制、服务请求处理以及Web服务集群的负载均衡等。Web服务器层向OGC应用服务器层发送服务请求，并将服务结果数据已SOAP获JSON得形式提供给服务展现层的客户端显示。

### OGC应用服务器层

OGC应用服务器层主要负责的是整个分布式OGC服务框架的后台服务的计算逻辑和数据存取逻辑，如图2.1，主要包含三个部分：OGC应用服务器集群、分布式缓存集群、集群管理监控模块，起到承上启下的作用。承上连接Web服务器层,为Web服务器层提供数据和计算服务，在Web服务器层的实现中，Web服务器组件是通过调用OGC应用服务器组件的本地代理来获取相应数据的。而代理真正的实现体在OGC应用服务器层，二者通过Ice中间件进行传输；启下利用数据引擎层，为各个不同的OWS服务提供相应地数据和处理操作；

同时OGC应用服务器层还承担了服务组件的管理、部署和整个应用服务器集群的监控和负载均衡。主要业务功能点包括WMS服务、WFS服务、WCS服务、WMTS服务、目录与元数据服务、打包下载服务 。前4种服务是标准的OGC服务，后2种服务是扩展的业务服务。

Web地图服务（WMS）：可以动态地利用地理信息中的空间数据绘制地图。地图本身并不是地理数据，而是数据的表现形式。Web地图服务所产生的地图一般是PNG, GIF或JPEG格式的，但偶尔也产生矢量格式的地图如：SVG或WebCGM格式的地图。根据OGC规范，地图服务是专门提供共享地图数据的服务，负责根据客户程序的请求，提供地图图像、指定坐标点的要素信息、以及地图服务的功能说明信息。这个规范定义了三个操作：GetCapabitities返回服务级元数据，它是对服务信息内容和要求参数的一种描述； GetMap返回一个地图影像，其地理空间参考和大小参数是明确定义了的；GetFeatureInfo（可选）返回显示在地图上的某些特殊要素的信息

Web要素服务（WFS）：通过WFS服务提供数据服务的情况下，任何使用Web服务的应用程序均可访问地图或地理数据库中的地理要素。与返回地图图像的 OGC Web地图服务 (WMS) 不同，WFS服务返回的是带有几何和属性信息的实际要素，客户端可以将这些要素与属性用于任何类型的地理空间分析。WFS服务也支持过滤器，由此用户可以在数据上执行空间查询和属性查询。WFS定义了五个操作：GetCapabilites返回Web要素服务性能描述文档（用XML描述）；DescribeFeatureType返回描述可以提供服务的任何要素结构的XML文档；GetFeature为一个获取要素实例的请求提供服务 。

Web地理覆盖服务（WCS）：提供的是包含了地理位置信息或属性的空间栅格图层，而不是静态地图的访问。根据HTTP客户端要求发送相应数据，包括影像，多光谱影像和其它科学数据。通WCS服务获取的栅格数据集被称为 coverage。网络覆盖服务由三种操作组成：GetCapabilities，GetCoverage和DescribeCoverageType。GetCapabilities操作返回描述服务和数据集的XML文档。网络覆盖服务中的GetCoverage操作是在GetCapabilities确定什么样的查询可以执行、什么样的数据能够获取之后执行的，它使用通用的覆盖格式返回地理位置的值或属性。DescribeCoverageType操作允许客户端请求由具体的WCS服务器提供的任一覆盖层的完全描述。

Web地图切片服务（WMTS）：是一种在web上使用缓存图像切片提供数字地图的服务，对客户端只提供这些预先定义好的单个瓦片的服务，将更多的数据处理操作如影像切割、图层叠加等放在客户端，从而缓解 GIS 服务器端数据处理的压力，改善用户体验。WMTS 服务􏰁供三种操作: GetCapabilities、GetTile 与 GetFeatureInfo。作用分别是返回元数据信息和获取地图切片。

目录与元数据服务：提供了一种数据发现服务，数据存储系统中存储多种不同类型的地图数据，面向不同的服务。目录服务可以将已经配置好的可以向外发布的数据源按照xml目录树的形式服务给上层，元数据服务是对目录服务的一种补充，目录服务提供的仅仅是简单的数据源的名称和所在的目录结构，元数据服务可以提供每一种服务出去的数据源的更详细的元数据信息给上层。

### 数据引擎层

如图2.1，数据存储系统是整个OGC服务系统的数据中心，为了适应不同服务不同数据的需要，整个OGC服务系统并不是用的单一的某一种数据库存储系统，目前是下面四种数据源组成：Oracle框架数据源、Sqlite瓦片数据源、本地文件系统数据源、Hadoop云端数据源。主要的两种存储介质为Oracle框架数据源和Hadoop云端数据源。Oracle数据源存储了各种类型的框架数据，包括地理空间的矢量数据、要素数据，Hadoop云端数据源存储了预先生成好的地图切片，按照瓦片图层来存储，每个瓦片图层同时存储了相应地元数据信息，之所以使用分布式的Hadoop云端存储，是因为WMTS服务的地图切片数据会非常大，单机数据库存储已无法满足，而HBase是一个针对结构化数据的可伸缩的、分布式和面向列的动态数据库，利用Hadooop集群来存储数据，可以有更好的容量扩展。本地数据源和Sqlite数据源是为了是在WMTS服务的数据量较小的时候方便更简单的演示整个服务的功能。

数据引擎层是用于存储、处理和保护数据的核心服务层。面对底层多种不同源的数据存储中心，若OGC应用服务器层直接对这么多不同种类的数据源进行存取操作，将会大大提升对OGC应用服务器层的开发难度，根据”分层解耦”的思想，数据引擎层将完全屏蔽底层多种存储源，对上层暴露一个统一的存取接口。

## 管理监控子系统

OGC应用服务器层是分布式的集群服务，服务的部署、管理相比较单机服务要复杂的多，对性能和稳定的要求也更高。管理监控子系统是OGC应用服务器层的辅助模块，面向的用户的服务管理员。主要有两个部分组成：集群监控模块和集群管理模块，两个模块的具体功能如下。

### 集群监控模块

集群监控模块的功能是监控OGC应用服务集群每一个服务器上和服务器上的每一个服务组件，包括服务集群里每个服务器的状态、ip、负载情况；服务集群中每个服务器上每个服务组件的状态、被调用的次数等。

### 集群管理模块

集群管理模块的功能是负责OGC应用服务器集群的控制，实现以下功能：添加、删除服务器节点；启用、关闭单个服务器(或整个集群)的某个服务组件；配置整个服务集群上某个负载组件的负载均衡策略。

管理子系统的是部署在OGC应用服务集群的中心节点上，两个功能模块实际是一体的，隶属于一个管理监控子系统的设计实现中。

基于上述的架构和功能点，本文重点是描述OGC应用服务器层的设计和实现过程，以及相应得管理监控子系统的设计和实现。在分布式对象中间件Ice的IceGrid基础上实现分布式集群服务功能。

## 相关技术分析

### WebGIS与OGC Web服务规范

WebGIS是一种基于网络的地理信息系统，结合了Web技术，GIS技术和数据库技术，其核心是在GIS中嵌入HTTP标准的应用体系，实现Internet环境下地空间信息管理和发布。WebGIS通过互联网对地理空间数据进行发布和应用，以实现空间数据的共享和互操作，如GIS信息的在线查询和业务处理等。WebGIS可以采用多主机、多数据库进行分布式部署，通过Internet实现互联，是一种浏览器/服务器(B/S)模式，服务器端向客户端提供信息和服务，浏览器端（客户端）具有获得各种空间信息和应用的功能，WebGIS主要作用是进行空间数据发布、空间查询与检索、空间模型服务、Web资源的组织等。

WebGIS是通过HTTP协议来进行请求的发送和接收返回结果，且遵循SOAP协议，WebGIS结构应用已经由浏览器/网络服务器/数据服务器（Browser/Web Server/Data Server）三层架构阶段进入到浏览器/网络服务器/应用服务器/数据服务器（Browser/Web Server/Application Server/Data Server）四层架构阶段。在新的四层架构中，网络服务器和应用服务器分离，并且其间还可以插入二次开发和扩展功能，其中的应用服务器一般为支持远程调用的组件式GIS平台，或由组件式GIS平台封装而成。将GIS复杂数据分析与处理功能（编辑、拓扑关系的构建、对象关系的自动维护、制图）移到GIS应用服务器上，使客户端与服务端的数据传输减少到最少的程度，为在Internet上实现复杂、大规模的地理信息服务提供了可能。这一架构带来的巨大优势是使服务器端具有极强的扩展性，因此作为应用服务器的组件式GIS所具备的功能，都可以通过B/S结构实现，WebGIS不再是只能满足地图浏览和查询的简单软件了，而是一个体系先进，功能强大的服务器端GIS（Server GIS）。新的服务器端GIS将是未来应用的发展的主流。服务体系如图2.2：



图2.3 WebGIS四层架构模型

开放地理空间信息联盟（Open Geospatial Consortium-OGC）是一个非盈利的志愿的国际标准化组织，引领着空间地理信息标准及定位基本服务的发展目前在空间数据互操作领域，基于公共接口访问模式的互操作方法是一种基本的操作方法。致力于提供地理信息行业软件和数据及服务的标准化工作，致力于提供地理信息行业软件和数据及服务的标准化工作。

通过国际标准化组织（ISO/TC211）或技术联盟（如OGC）制定空间数据互操作的接口规范，GIS软件商开发遵循这一接口规范的空间数据的读写函数，可以实现异构空间数据库的互操作。基于http（Web）XML的空间数据互操作是一个很热门的研究方向，主要涉及Web Service的相关技术。OGC和ISO/TC211共同推出了基于Web服务（XML）的空间数据互操作实现规范Web Map Service，Web Feature Service，Web Coverage Service以及用于空间数据传输与转换的地理信息标记语言GML。OGC提出了一个能无缝集成各种在线空间处理和位置服务的框架即OWS(OGC Web Service)，使得分布式空间处理系统能够通过XML和HTTP技术进行交互，并为各种在线空间数据资源、来自传感器的信息、空间处理服务和位置服务，基于Web的发现、访问、集成、分析、利用和可视化提供互操作框架。

### 分布式对象中间件Ice

1964年，IBM发布OS/360操作系统，软件与硬件分离，同时，软件成为一个独立的产业正式登上产业界的舞台。中间件就是软件产业不断发展过程中自然产生的。中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源。中间件位于客户机/ 服务器的操作系统之上，管理计算机资源和网络通讯。是连接两个独立应用程序或独立系统的软件。相连接的系统，即使它们具有不同的接口，但通过中间件相互之间仍能交换信息。执行中间件的一个关键途径是信息传递。通过中间件，应用程序可以工作于多平台或OS环境。

在使分布式计算能为应用开发者所用的进程中,面向对象中间件是十分重要的一步 。开发者第一次拥有了这样的可能:不必是一个网络古鲁 (guru),就可以构建分布式应用——中间件平台会照管大部分网络杂务,比如整编 (marshaling)和解编(unmarshaling)(对数据进行编码与解码,以进行传送)、把逻辑对象地址 映射到物理传输端点、根据客户和服务器的原生机器架构改变数据的表示,以及应需自动启动服务器。

Ice 是ZeroC公司开发的一种面向对象的中间件平台。从根本上说,这意味着Ice 为构建面向对象的客户-服务器应用提供了工具、API和库支持。Ice应用适合在异种环境中使用:客户和服务器可以用不同的编程语言编写,可以运行在不同的操作系统和机器架构上,并且可以使用多种网络技术进行通信，支持Linux、Windows、Solaris和Mac OS X等主流的操作系统，支持C++、Python、Java、JS、.NET、PHP、Ruby等语言。无论部署环境如何,这些应用的源码都是可移植的。

服务端和客户端，即使是不同的操作系统，不同的编程语言，通过对象中间件Ice也可以完成通信。对象中间件Ice的本质是远程过程调用（Simple Object Access Protocol，RPC）框架。客户端和服务器端的通信协议，Ice中间件使用一种接口描述语言(Interface Description Language，IDL)Slice进行定义的，Slice是一种用于对象接口与其实现相分离的基础性抽象机制。Slice在客户端和服务器之间建立合约，描述应用所使用的各种类型及对象接口。Slice有自己的一套描述规则，这种描述与实现的具体语言类型无关，它是一种春岁的描述性语言，描述的是接口和类型（不是实现），所以无法用Slice编写可执行语句。用Slice语言完成接口的定义之后，用Ice提供的编译工具，将其映射成特定的语言，如C++、Java、Python等。在翻译之后，Ice服务接口被映射为纯虚的（只有类和接口的定义，没有具体实现）代理类和骨架类，二者都包含了Slice接口描述文件中声明的方法。对于客户端来说，可以调用代理类中定义的方法；对于服务器的实现来说，服务器端必须包含骨架类的代码。

上述代理类和骨架类都是纯虚类，服务器端无法真正为客户端提供服务，服务器端要通过继承骨架类形成Ice servant（服务体），完成骨架类中每个方法具体的实现。客户端调用本地的Ice服务代理（proxy）类完成客户端逻辑；服务器端的开发人员则专注于Ice servant类的开发来实现接口的功能。客户端调用Ice proxy类的方法，实际是服务器端的Ice servant类在提供服务，这中间的数据通信是Ice运行时负责，Ice远程过程调用RPC的本质仍然是传输层的网络通信，Ice运行时负责客户端与服务器之前网络通信的过程、负责数据的序列化和反序列化。客户端调用Ice proxy的方法，入参是结构化的数据，Ice运行时负责将其进行序列化，以字节流的形式传输到服务器端，服务器端将接收到的字节流在反序列化成结构化的请求参数，并定位到指定的Ice servant进行请求处理，所得到的结果数据再通过Ice 运行时返回给客户端。基于Ice中间件搭建的开发框架的结构图如图2.3：



图2.4 Ice中间件平台的C/S架构

例如，如图2.4的发开场景，客户端使用服务器端的打印功能，客户端采用Java语言开发，服务器端采用C++语言开发，二者通过Print.ice描述通信接口。



图2.5 基于Ice中间件搭建服务的开发框架

当采用Ice中间件实现异构的跨语言的RPC的服务方案时，首先由接口定义人员使用Slice语言进行接口定义Printer.ice，以构建通信接口Ice打印对象。之后，对于同一份Slice接口文件Printer.ice，服务器端开发者通过slice2cpp编译其生成C++骨架类代码，而客户端开发者则通过slice2java编译其生成Java代理类代码。最后，两者分别继承可调用代码中Ice打印对象所映射的纯虚类形成本地类型，从而为上层应用的RPC调用关系完成开发框架的搭建。

### 负载均衡

由于目前现有网络的各个核心部分随着业务量的提高，访问量和数据流量的快速增长，其处理能力和计算强度也相应地增大，使得单一的服务器设备根本无法承担。在此情况下，如果扔掉现有设备去做大量的硬件升级，这样将造成现有资源的浪费，而且如果再面临下一次业务量的提升时，这又将导致再一次硬件升级的高额成本投入，甚至性能再卓越的设备也不能满足当前业务量增长的需求。针对此情况而衍生出来的一种廉价有效透明的方法以扩展现有网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性的技术就是负载均衡（Load Balance）。

负载均衡，多指的是对访问服务器的负载进行平衡措施，英文名称为Load Balance，其意思就是分摊到多个操作单元上进行执行，例如Web服务器、FTP服务器、企业关键应用服务器和其它关键任务服务器等，从而共同完成工作任务。建立在现有的网络结构之上，它提供了一种廉价有效透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性，使得服务器即使在负载很重的情况下，仍保持快速响应的能力。

负载均衡的调度策略方法是指对用户请求进行分配调度时所采用的决策方法。负载平衡的调度决策方法选取是否恰当将会直接影响到负载平衡方案的效率。调度决策方法可以分为加权法、随机法、轮转法、硬件负载平衡器、散列法、最小连接法、最快响应法7种方法(引自 201001网络负载平衡技术及其在WebGIS中的应用.pdf)。负载均衡的具体实施方法如下：

HTTP重定向，相信对于所有web程序员都不陌生，例如我们请求某个页面时，被转向登录页，登录页面之后又被转到了某个页面。大致来说就是，浏览器请求某个URL后，服务器通过HTTP响应头信息中的Location标记返回一个新的URL，这样浏览器主会继续请求这个新的URL，完成自动跳转。也正是因为HTTP重定向有请求转移和自动跳转的能力，所以我们就可以用它来实现负载均衡以实现WEB扩展。

DNS负载均衡：我们知道DNS是负责解析域名的，当我们用域名访问站点时，实际上都会经过DNS服务器来获取域名指向的IP，实际上DNS服务器完成了域名到IP的映射，同样这个映射可以是一对多的，也就是DNS可以把对域名的请求按照一定的策略分配到不同的服务器上，这样我们就可以依此来实现负载均衡。貌似和HTTP重定向很样，但是实现的机制却是完全不同的。（在window下可以用nslookup命令查询域名对应的IP地址列表，这个命令会返回离你最近的DNS服务器缓存的记录并不一定是全部）

反向代理负载均衡：它同样可以作为调度器来实现负载均衡系统。反向代理服务器核心工作是转发HTTP请求，它工作在HTTP层面，也就是TCP七层中的第七层应用层，因此基于反射代理的负载均衡系统也称为七层负载均衡。目前几乎所有主流WEB服务器都支持基于反向代理的负载均衡，因此实现它并不困难。

IP负载均衡：基于IP负载均衡的系统工作在传输层，会对数据包中的IP地址和端口信息进行修改，所以也称为四层负载均衡。它会在数据到达应用层之前，已完成转发，因些这工作都是由系统内核来完成，应用程序对此无能为力，当然性能来说也会很大的提升。

直接路由：不同于IP负载均衡，直接路由负载均衡高度器工作在数据链路层，它通过修改数据包的目标MAC地址，将数据包转发到实际服务器，不同的是，这些处理的结果直接发送给用户，不再经过调度器。这时我们的实际服务器必须直接连接到外网，并且不在以调度服务器为默认网关。

IP隧道：就是调度器将收到的数据包封装到一个新的IP数据包中，转交给实现服务器，然后实际服务器可以处理数据包直接响应客户端。

# 高性能OGC应用服务器设计

## OGC应用服务器层的设计

### 分布式请求

OGC应用服务器层主要负责的是整个分布式OGC服务框架的后台服务的计算逻辑和数据存取逻辑，起到承上启下的作用。承上连接Web服务器层，启下利用数据引擎层，实现了整个分布式OGC服务系统的核心功能。主要功能点包括各种不同类型的地图数据、要素数据、元数据的获取和传输；地图瓦片的实时渲染；瓦片的打包下载处理。

OGC应用服务器层的物理集群部署图如下3.1：



图3.1 OGC应用服务器集群物理部署

如图3.1，在物理部署方面，Ice注册中心是整个OGC应用服务器集群的中心节点，负责管理和配置整个服务器集群。中心节点知道整个OGC应用服务器集群中所有服务器的详细信息，所有的客户端（Web服务器层）的请求都是首先请求Ice注册中心获取一台应用服务器的代理信息，这里Ice注册中心可以根据不同的负载均衡策略来返回特定的应用服务器的代理信息，客户端再根据代理信息调用proxy的接口，Ice 运行时将此次调用定位到前面指定的Ice servant的接口上，Ice servant完成此次调用并将结果通过Ice运行时再返回给Ice proxy，服务的工作流程如下3.2：



图3.2 应用服务器的工作流程

应用服务请求的路由方式分为两种：直接请求和间接请求。直接请求是Web客户端直接请求应用服务器集群中某应用服务器节点，间接请求如上所述，Web客户端需要首先请求一次Ice注册中心获取集群内某应用服务器节点的代理信息，然后再根据代理信息去请求应用服务器节点。直接请求的优点是简单快捷，但是缺点也很明显，首先：应用服务器集群的所有节点的信息必须都暴露给客户端，客户端的编程任务加重。然后：客户端还需要去关注应用服务器集群的状态信息，如应用服务器集群内某节点发生改变（失效或更新等），都会对客户端造成影响。最后：客户端自己选择应用服务器集群内某个应用服务器节点，就以为这负载均衡模块必须由客户端自己来负责，这是非常不合理的，客户端和服务器的耦合度过高，负载均衡是服务器模块，更不能把任务抛给上层。由于以上的缺点分析，在OGC应用服务器端的设计中，我们采用间接请求的方式。

### 模块设计

根据2.2.3章节所述，OGC应用服务器端需要向客户端提供WFS、WMS、WCS、WMTS、目录与元数据服务、打包下载服务 6种服务，每种服务类型设计了相对的slice接口描述文件。Web服务器层使用的是Java语言进行开发，通过slice2java将每种服务的slice文件编译生成对应的Ice proxy类与代理方法接口。OGC应用服务器的主要模块的设计是基于IceGrid完成，主要包含了应用服务组件和管理监控模块。

应用服务组件是部署在每个应用服务器节点上运行的，是完成OGC应用服务层主要功能的载体，实质上是封装了不同Ice servant服务类及其定义的方法的可执行文件。应用服务组件可以通过管理监控模块在每个应用服务器节点上进行动态的加载和卸载。应用服务器端的开发人员在进行服务组件开发时，如前面所述，首先是进行接口定义，再由slice2cpp生成相应地骨架类，根据骨架类里面的方法的定义完成所有方法的实现并生成相应地可执行文件。最后，将可执行文件通过管理模块配置到应用服务集群中去。

管理监控模块主要负责OGC应用服务器集群里面每个应用服务组件的部署、配置、监控和记录跟踪，这一模块将会在第五章中做详细描述。

## OGC应用服务器层的网络框架模型

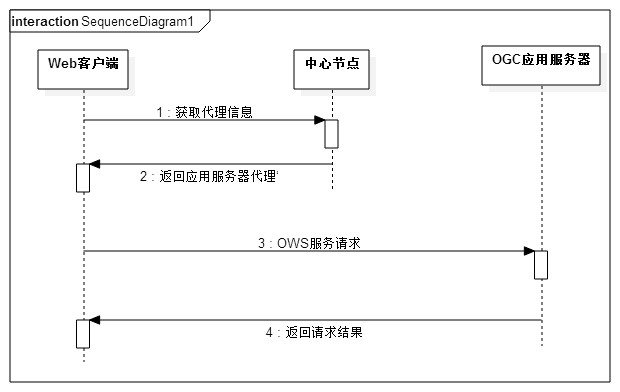
在网络编程中，阻塞、非阻塞、同步、异步经常被提到。同步和异步是针对功能的执行顺序来说的，而阻塞和非阻塞是针对等待IO数据的方式说的。因此这是两对概念，同步与阻塞，异步与非阻塞没有必然的联系。同步就是工作线程在处理IO时等待IO完成再继续后面的工作；异步就是工作线程不等待IO处理的结果就继续后面的工作，而IO处理结果将通过回调的方式返回；阻塞是在等待IO时，如果IO没有可用数据或数据没有传送完成，那么一直等待下去，直到IO处理完数据再返回；非阻塞就是不管IO是否有可用数据或数据已经传送，照样返回。只要把同步和阻塞分清，这四个概念就很容易理清了。

同步、异步、阻塞、非阻塞等原理确定了网络通信的基本网络通信模型结构。它们处理的怎么等待数据和怎么收发数据的问题，特别是服务端，当成千上万的连接发生并发时，降低cpu的占用率，减少内存使用率，减少带宽就成为了很关键的问题。同步和异步关注的是消息通信机制 (synchronous communication/ asynchronous communication)。所谓同步，就是在发出一个"调用"时，在没有得到结果之前，该"调用"就不返回。但是一旦"调用"返回，就得到返回值了。换句话说，就是由"调用者"主动等待这个"调用"的结果。而异步则是相反，"调用"在发出之后，这个调用就直接返回了，所以没有返回结果。换句话说，当一个异步过程调用发出后，调用者不会立刻得到结果。而是在"调用"发出后，"被调用者"通过状态、通知来通知调用者，或通过回调函数处理这个调用。

在进行OGC应用服务器层开发阶段，基于迭代开发的思想，我相继用了四种网络模型：同步模型、异步模型、基于线程池的同步会话模型、基于线程池的异步会话模型。

### 同步模型

通过使用远程过程调用，远端服务调用变得像传统的方法调用一样容易，现代中间件技术力图减轻程序员在转向分布式应用开发时的负担：你调用某个对象上的方法，当该方法完成时，就会返回结果，或是抛出异常。当然，分布式系统中，对象的实现可能会驻留在另一个主机上，从而带来一些语义上的差异，程序员必须加以注意，比如远程调用的开销，以及可能发生与网络有关的错误。尽管有这样一些问题，程序员的面向对象编程经验仍然是有用的，上述的同步编程模型（发出调用的线程会阻塞到操作返回）是我们所熟悉和容易理解的，同步模型的时序图如图3.3：



3.3 同步模型的服务请求过程

同步模型的优点：编程实现简单，每一种服务的每一个请求调用就是一次远程过程调用，服务器端处理完此次调用直接将结果返回或抛出异常。

同步模型的缺点：同步模型下，服务器端的并发量收到限制，服务端Ice的每个接收请求的线程不仅要接受客户端发过来的调用请求，还要负责每次调用请求的处理逻辑。不仅客户端的调用请求会一直阻塞住等待返回，服务器端的接收线程在每一次调用过程中阻塞时间过长，当并发量和接收线程池内的线程数量相当时，接收线程资源很容易就被耗尽，从而无法满足更多的并发请求。

### 异步模型

Ice 在本质上是一个异步的中间件平台,出于对应用 (及其程序员)的考虑而模拟了同步的行为。当Ice应用通过代理、向远地对象发出同步的双向调用时,发出调用的线程会阻塞起来,以模拟同步的方法调用。在此期间，Ice run time 在后台运行,处理消息,直到收到所需的答复为止,然后发出调用的线程就可以解除阻塞,解编结果了。但是,在许多情况下,同步编程的阻塞本质太过受限。例如,在等待远地调用的响应返回时,应用本来可以另外做一些工作;在这样的情况下使用同步调用,应用就会被迫把工作推后、等待响应返回,或是在另外一个线程中执行该工作。如果这些做法都不可接受, Ice 的异步设施提供了一种有效的解决方案,可以改善性能和可伸缩性,或是简化复杂的应用任务。

一个服务器在同一时刻所能支持的同步请求数受到 Ice run time 的服务器线程池的尺寸限制。如果所有线程都在忙于分派长时间运行的操作,那么就没有线程可用于处理新的请求,客户就会经常处于不可接受的无响应状态。异步模型有两种概念: 异步方法调用和异步方法分派。

异步方法调用(AMI)这个属于描述的是客户端的异步编程模型支持。如果使用AMI发出远地调用，在Ice run time 等待答复的同时，发出调用的线程不会阻塞,相反，发出调用的线程可以继续进行各种搞活动，当答复最终到达时，Ice run time会通知客户端应用。

异步方法分派(AMD)，一个服务器在同一时刻所能支持的同步请求数受到 Ice run time 的服务器线程池的尺寸限制。如果所有线程都在忙于分派长时间运行的操作,那么就没有线程可用于处理新的请求,客户就会经常处于不可接受的无响应状态。异步方法分派（AMD）是AMI的服务器端等价物，能够解决这个可伸缩性问题。在使用 AMD 时,服务器可以接收一个请求,然后挂起其处理, 以尽快释放分派线程。当处理恢复、结果已得出时,服务器要使用 Ice run time 提供的回调对象,显式地发送响应。AMD操作通常会把氢气数据（也就是，回调对象和操作参数）放入队列中，供应用的某个线程（或线程池）随后处理用，这样，服务器就是分派线程的使用率降低限度，能够高效地支持数千并发客户。

利用Ice提供的异步编程模型，接口的定义和同步方式下几乎一样，唯一的区别是增加关键字 [“amd”]。应用服务器端的开发人员可以在两个层面批注上[“amd”]关键字，接口（interface）的层面或单个方法的层面。如果是在为一个接口进行指定异步属性，那么它的所有方法生成的代码都将会有异步支持，如果只有某些操作需要异步支持，那么可以只为这些操作前面制定[“amd”]关键字，从而是生成的代码的数据降到最低限度。在2.2.3章节中介绍了我们所要实现的6中服务的介绍，例如WFS服务的接口定义如下:

["amd"] interface WFS{

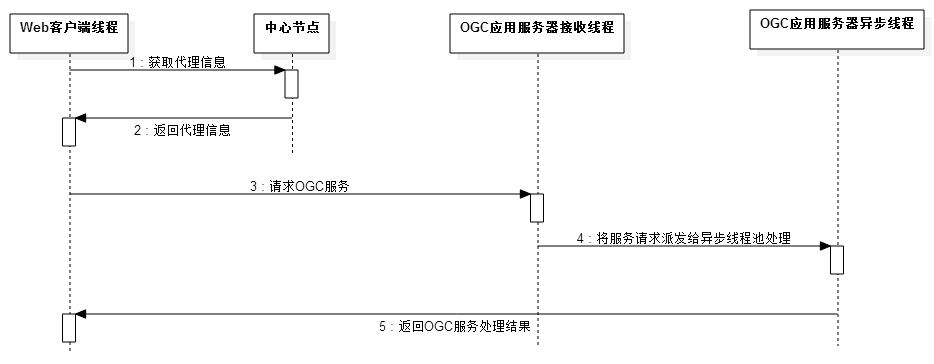
FileInfo getCapabilitiesFile();

FileInfo describeFeatureTypeFile(string typeName);

FileInfo getFeatureFile (string typeName, BBOX box, stringStream propertyName);

};

在WFS接口前增加了Ice的异步关键字，使得WFS服务的内部的三种方法均为异步方法。异步模型将服务器端Ice运行时的分派线程的处理任务减轻，将请求的分派和请求的处理分开来，大大提高了应用服务器端的并发能力。AMD操作的逻辑实现分为两种线程：分派线程（接收调用的线程），以及响应线程（发送响应的线程）。分派线程仅仅负责接收客户端发过来的请求和请求参数，响应线程负责具体的运算逻辑和请求结果的响应。通过把费时的请求交给响应线程去处理，servant可以改善可伸缩性，并节省资源，从而提高并发性。异步编程模型的要从线程处理的角度看流程变化，如图3.4为异步模型的时序图：



3.4 异步模型服务请求过程

但是异步模型并非能解决所有应用服务器端开发遇到的场景，它也有这自己的局限性。1）响应线程池无法动态的调整，服务器端的并发量并不是一个稳定的常量，响应线程池如果开得太小，那么很多请求会得不到响应，服务器端会发生阻塞，导致更多的请求得不到响应；响应线程池如果开得太大，又会造成资源浪费，线程资源过多，线程之间的调用切换也会占据一部分时间，甚至可能倒置服务器的性能下降。2）响应线程池对客户端发送过来的请求是没有优先级处理，全部都是按照时间顺序以竞争的方式获取来获取请求任务类完成的，这种处理方式简单快捷，但却不灵活。因为不同服务的不同操作的优先级并不一定是相同的，有些操作客户端需要得到快速响应以便进行下一步操作，如各种OWS服务的Getcapabilities方法；有些操作是I/O密集型，需要比较长的处理时间，可以适当的调低其优先级，如打包下载服务接口。单纯的异步模型无法满足给不同服务的不同操作设置优先级。3）无论是同步模型还是异步模型，当请求操作处理完毕之后，所得到的结果都是一次性给客户端返回的，Ice运行时的通信方式仍然是基于传输层的网路通信，Ice对每次通信的字节流长度也是由限制的，当服务请求的结果数据量大小超过这个限制时，就无法正常的返回。例如WFS的GetFeature操作的返回结果可能是几百兆的feature地理数据，打包下载服务的打包后的瓦片集能达到上G的大小，这些数据量无法通过一次性的网络传输返回。

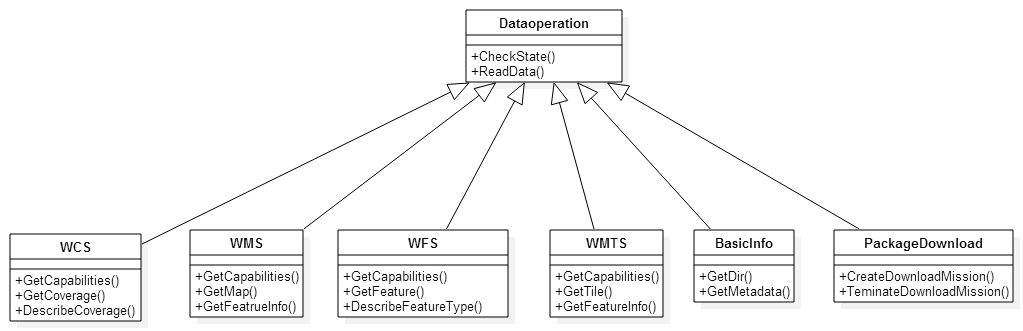
基于以上的三点局限性，异步模型无法完全的解决OGC应用服务层各种组件的设计开发，下面将介绍一种基于线程池的会话模型，会话模型也是由两种，同步会话和异步会话。

### 同步会话模型

由异步模型的局限性，我们用一种新的网络框架模型。一次网络服务的请求如果局限在一次处理并返回，无论是使用还是异步模型，都无法突破Ice运行时一次传输数据流大小的限制。此时必须要讲一次网络服务请求进行分解，请求任务和获取结果至少需要分成两个步骤，如此一来，客户端发送一个请求任务到服务器端，服务器端处理完毕之后，客户端再请求一遍将结果数据拿走，注意，这里是可以循环多次获取结果数据的，如此一来就可以解决结果数据大小的限制。

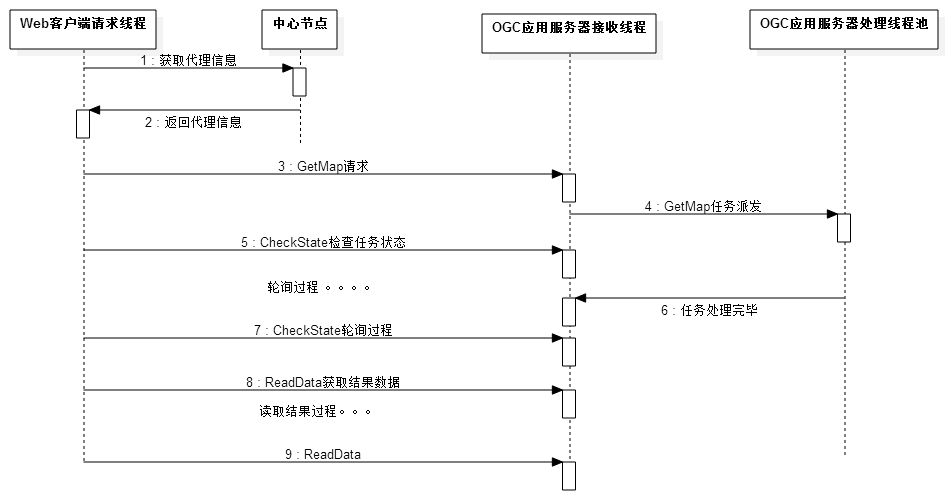
但是，客户端发送请求任务之后要，需要确定服务器端的服务请求已经处理完毕才能进一步发送获取数据的请求。如此一来，当服务器端的请求处理比较耗时的时候，客户端就会一直等待下去得不到响应，Web客户端和OGC应用服务器之间的通信方式是基于Ice中间件的TCP进行传输，客户端长时间得不到响应，甚至会出现请求超时的情况，导致连接失效。

为了不让客户端的请求处于长时间等待状态，我们采取的解决方案是模仿http协议的session。http是无状态的协议，客户每次读取web页面时，服务器都打开新的会话，而且服务器也不会自动维护客户的上下文信息，那么要怎么才能实现会话跟踪呢？session就是一种保存上下文信息的机制，它是针对每一个用户的，变量的值保存在服务器端，通过SessionID来区分不同的客户,session是以cookie或URL重写为基础的，默认使用cookie来实现，系统会创造一个名为JSESSIONID的输出返回给客户端Cookie保存。我们服务的Web客户端和OGC应用服务器之间的通信也是无状态的，为了维护客户端和服务器端一次服务请求的状态，我们用类似于session功能的uuid来唯一标识某一次服务请求的全过程，我们将客户端原本一次的服务请求分解成三次会话过程：1）客户度发送请求任务，应用服务器端生成此次请求任务的唯一标识uuid返回给客户端。2）客户端根据uuid向应用服务器查询请求任务的准备状态。如果请求任务准备完成，则进入第三步，否则返回第二步。3）当服务器端的请求任务完成，结果数据已经准备好，则客户端根据uuid向服务器端获取结果数据。根据服务请求的三次会话的过程，设计的服务类图如下3.5:



3.5 同步会话模型设计类图

如类图所示，每种服务都继承了公共接口DataOperation，该接口提供了两种公共函数，CheckState和ReadData，分别用来检查应用服务器端服务请求的准备状态和从应用服务器端读取请求结果数据。用WMS服务的GetMap操作来举例，时序图如下3.6：



3.6 同步会话模型的服务请求流程

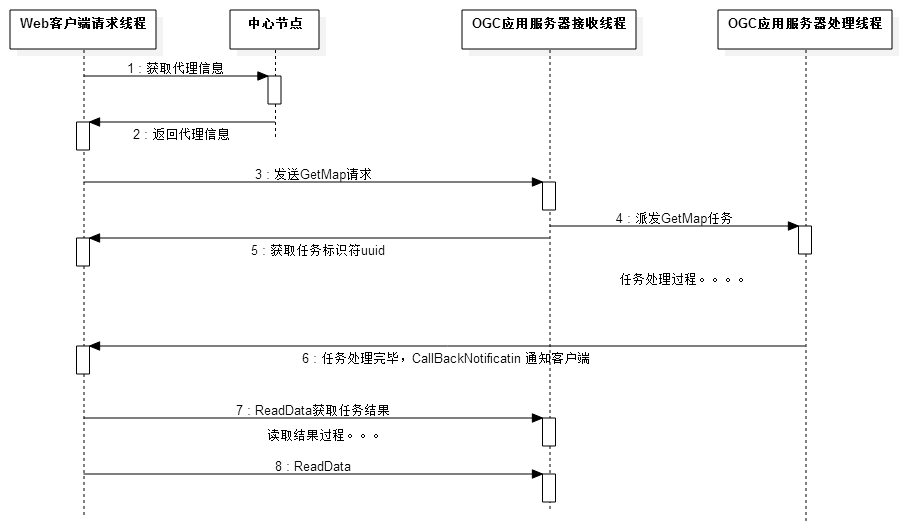
上述的网络框架模型是从客户端的流程来分析，而从OGC应用服务器的角度分析来看。应用服务器端接收Web客户端发过来的服务请求之后，生成一个请求任务，任务中包含任务的唯一标识符uuid，请求类型，请求参数，请求状态，并将任务放入任务队列之中，然后将任务的uuid返回给客户端。服务器端有一个处理任务用的线程池，线程池中线程已竞争的方式从任务队列中获取任务并执行；线程拿到一个任务执行完毕之后将结果数据放入结果队列之中，并将该任务的状态设置成可读状态。待到客户端将结果队列中此次任务的结果数据全部获取完毕之后，服务器端线程将此次的结果数据删除（至于缓存策略，会在第五章详述）。

以上的基于线程池的同步会话模型适用于各种服务场景，结果数据无论多大，分多次就可以读取完毕。但是同样有着一个缺点：Web客户端频繁得去轮询应用服务器端的服务请求状态，这样的轮询操作也浪费了带宽和连接资源。对于Web客户端来说，需要不断得去调用CheckState方法去查询服务器端状态，占用客户端的cpu资源，无法处理其他任务，不利于Web客户端的并发性；对于OGC应用服务器端来说，Web客户端的每次CheckState请求，服务器端也都要做出相应地响应，Ice运行时的派发线程接收到查询请求，会从任务队列中获取此次任务的状态信息，并返回给Web客户端。这一过程不仅占用应用服务器的cpu资源，同时也占用了派发线程的资源，如果大量Web客户端进行并发的轮询请求，派发线程可能会被这些轮询请求占满，从而无法为新的Web客户端的任务请求进行服务，限制了整体服务框架的并发性。

### 异步会话模型

在设计模式中有一条设计原则叫做好莱坞原则——“不要打电话给我，我会主动给你打电话”。根据好莱坞原则，我们基于Ice的异步编程模型改进了上述的基于线程池的同步会话模型，采用异步通知机制来通知Web客户端任务请求在应用服务器端的状态。改进后，Web客户端既可以按照之前的同步会话模型去通过CheckState轮询应用服务器的请求处理状态，也可以什么都不管，继续执行自己的操作，等待应用服务器端的显示回调通知。异步回调通知的流程如下：1）Web客户端发送服务操作请求给OGC应用服务器端并获取该次请求的uuid，然后继续进行自己的其他逻辑操作。2）OGC应用服务器端的线程池处理此次任务请求，当处理完毕后将结果放入结果队列中，并显示的回调Web客户端，通知其任务准备完毕。3）Web客户端接收到应用服务器的回调通知之后再去想应用服务器去获取此次任务的结果数据。

异步会话模型需要应用服务器端的回调通知机制，我们将基类中再添加回调方法CallBackNotification，该方法为服务器端回调客户端用，任然利用WMTS的GetMap请求举例，异步会话的时序图如下3.7：



3.7 异步会话模型的服务请求流程

基于线程池的异步会话模型解决了Web客户端不断轮询造成的网络通信负担，Ice运行时的派发线程池单纯处理请求任务的派发工作，应用服务器层的cpu负担减轻，提升了服务整体的并发能力；Web客户端不用再去向应用服务器进行轮询操作，可以让请求线程挂起或处理其他任务，大大减轻Web客户端的负担。但是，基于线程池的异步会话模型有这些优点的同时，其缺点也是不能忽略。客户端和应用服务器端的开发难度加大，异步编程模型相较于同步编程模型来说虽然更为高效但也更为复杂。

## 高效的资源管理

### 概述

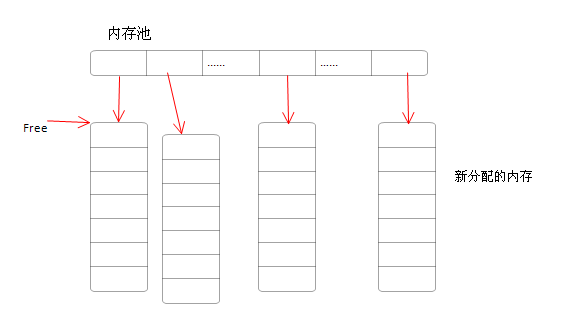
OGC应用服务器层是整个分布式OGC服务框架的核心，作为后台服务，对稳定性、效率和资源利用率的要求非常高。

Pool（池）的概念被广泛的应用在服务器端软件的开发上。使用池结构可以明显的提高你的应用程序的速度，改善效率和降低系统资源的开销。所以在现在的应用服务器端的开发中池的设计和实现是开发工作中的重要一环。那么到底什么是池呢？我们可以简单的想象一下应用运行时的环境，当大量的客户并发的访问应用服务器时我们如何提供服务呢？我们可以为每一个客户提供一个新的服务对象进行服务这种方法看起来简单，在实际应用中如果采用这种实现会有很多问题，显而易见的是不断的创建和销毁新服务对象必将给造成系统资源的巨大开销，导致系统的性能下降。针对这个问题我们采用池的方式。池可以想象成就是一个容器保存着各种我们需要的对象。我们对这些对象进行复用，从而提高系统性能。从结构上看，它应该具有容器对象和具体的元素对象。从使用方法上看，我们可以直接取得池中的元素来用，也可以把我们要做的任务交给它处理。

### 内存池

通常我们习惯直接使用new、malloc等API申请分配内存，这样做的缺点在于：由于所申请内存块的大小不定，当频繁使用时会造成大量的内存碎片并进而降低性能。对于我们的OGC应用服务器层的开发来说，需要频繁得申请使用内存资源。如果直接使用new、malloc等API申请分配内存，会产生大量的内存碎片，从而使得服务程序的运行效率越来越低。

内存池(Memory Pool)是一种内存分配方式。内存池则是在真正使用内存之前，先申请分配一定数量的、大小相等(一般情况下)的内存块留作备用。当有新的内存需求时，就从内存池中分出一部分内存块，若内存块不够再继续申请新的内存。这样做的一个显著优点是，使得内存分配效率得到提升。在服务中，我们通过改进SGI STL中的内存分配器（allocator）来设计内存池。SGI STL的内存分配器是单线程版本的，且内存池中每个内存的节点数量无法动态增减。我们的OGC应用服务器端设计的线程池是要在多线程环境下工作，并且最好能根据应用服务器层的需求情况，进行动态的增减。设计图如下3.8：

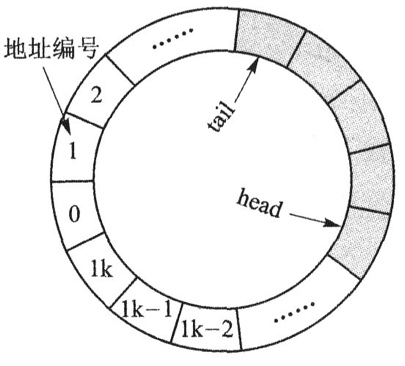


3.8 内存池

如图：内存池中有一个桶（数组），桶内部的每个节点是一定大小内存块链表的头结点，当应用服务器的某个线程需要内存资源的时候，便向内存池调用get操作，获取一个适合大小的桶节点的内存块，使用完毕之后再调用put将内存块放回到内存池的桶内部。为了让内存池线程安全，在桶内的每个节点都放上互斥锁保证资源的安全。同时，为了让内存的尺寸适合应用服务器的需求变化，会有后台线程每隔两分钟检测内存池剩余内存块情况，以此为判断动态得向内存池中添加新的内存块资源或消除多余资源。

### 环形内存缓冲区

OGC应用服务器层处理服务请求结果的方式是将请求结果直接放入结果队列之中。如此一来，请求结果的内存数据必须全部准备完毕才能通知Web客户端过来读取此次请求结果。但是WFS的GetFeature操作的结果数据有达到几百兆甚至上G大小，这样大规模的数据量，不可能全部存储在内存里然后给客户端读取。一方面，应用服务器的内存资源有所限制，另一方面，如此大的数据量，应用服务器的处理也是非常耗时，客户端等到应用服务器全部处理完毕再读取数据导致流程串行化，客户端等待时间太长。为了解决上述问题，我们设计了环形内存缓冲区，结构图如下3.9：



3.9 环形缓冲区结构图

如图，环形内存缓冲区中有lk个组成环形的内存块节点组成，有读游标和写游标用来标识缓冲区中内存数据情况，应用服务器的处理线程在获取结果数据期间，不断向环形内存缓冲区中写入结果数据，并通知Web客户端过来读取这一部分数据，Web客户端接收到通知之后，便从环形内存缓冲区中将该部分结果数据取走。应用服务器的处理线程作为生产者，Web客户端的读取请求线程作为消费者，二者共享该环形内存缓冲区资源。

### 数据库连接池

Pool（池）的运用如上我们已经在应用服务器中设计了线程池和内存池，各个服务在使用数据引擎层与数据存储系统进行数据交互，二者交互的媒介是数据库句柄。数据库句柄的创建和销毁是需要占用网络资源，如果每一个需要与数据库交互的线程都需要创建数据库句柄，使用完毕之后销毁，这是相当耗时且影响应用服务器性能的做法。利用池的思想，我们设计了数据库连接池，是一个可以动态增长的存取数据库连接资源的队列。

在服务器开启时，应用服务器主动创建一部分数据库句柄资源放入数据库连接池中，在服务过程中，处理线程需要数据库句柄资源可以直接想数据库连接池获取，使用完毕之后再放回连接池中。同时，数据库句柄是面向网络的资源，意味着如果长时间没有客户端使用，该网络连接可能会断开，连接资源失效。类似于内存池的后台维护线程，数据库连接池同样有后台连接线程，每个一段时间会检测连接池内所有数据库连接的有效性，对失效的数据库连接进行重新连接，并动态得调整数据库连接池的大小。

# 集群管理监控与分布式缓存模块设计

## 概述

OGC应用服务器层主要包含了OGC应用服务器群、分布式缓存集群、集群管理监控模块，如图4.1。在第三章中，我们详述了如何设计高效地OGC应用服务组件，从高效的网络框架模型和资源管理策略提升了单个应用服务器节点的并发能力和稳定性。但是无论单个应用服务器节点的性能如何得提升，也显然无法满足高并发压力的需求，OGC应用服务器层是整个分布式OGC服务系统的核心层，所以我们采用多台服务器协同工作，用来提高整个分布式OGC服务系统的并发处理能力。



4.1 OGC应用服务层器框架

但是如何在不同的OGC应用服务器节点之间进行协调工作，实现合理的请求任务分配，使集群不会出现一台服务器过于忙碌，而其他的服务器节点却没有充分发挥处理能力的情况，负载均衡是一种分布式系统框架下的业务调度和负载分配的解决方案，目的是为了使分布式系统中的每台服务器节点都能够最大限度的发挥功效。

管理监控模块是对应用服务器集群的辅助模块，用来支持应用服务组件的动态部署、控制应用服务器集群和获取应用服务器集群的信息，方便服务器管理人员更好的管理整个OGC应用服务器层。分布式缓存模块同样是OGC应用服务器层的一部分，缓存模块的存在大大降低了应用服务器的计算负载，对整个服务器的并发性能的提升起到了重要作用。

## 负载均衡

### 负载分配模型

OGC应用服务器集群的负载均衡我们采用了两级负载均衡策略，分别是在IP层基于NAT实现的虚拟服务器负载均衡和在应用层基于IceGrid框架实现的负载均衡。OGC应用服务器集群部署如图4.2：



4.2 OGC应用服务器集群物理部署

OGC应用服务器是基于IceGrid实现的分布式服务，3.1章节中详述了分布式请求的过程，传统的IceGrid实现的分布式服务均是一个中心节点对应多个服务器节点，所有的客户端请求都是通过中心节点实现派发。如此一来，中心节点边成为单点，虽然中心节点的业务逻辑十分简单，仅仅需要更具负载均衡策略选择服务器集群中的一台服务器即可，但是，随着集群规模的增大，并发量的增大，一台中心节点将见见无法满足需求，而且，当中心节点崩溃时，整个服务器集群都将无法向客户端提供服务。

一级负载均衡面向的是OGC应用服务器集群的中心节点，目的是解决中心节点的单点问题。如图4.2，应用服务器集群有多个中心节点，Web客户端通过一级负载均衡策略从多个中心节点中选择一个，然后向该中心节点发出请求，获取应用服务器的代理信息，再根据该代理信息请求服务。中心节点的数量随着应用服务器节点数量的增加也会动态增加，例如我们配置1台中心节点对应10台应用服务器节点，保持中心节点和应用服务器节点数量是1:10的关系，如此一来，各个中心节点之间互不干扰。

二级负载均衡是对OGC应用服务器节点的负载均衡，利用IceGrid框架来完成中心节点和应用服务器集群的开发。如图4.x，应用服务器节点利用IceGrid框架定期想中心节点发送心跳数据包，应用服务器节点定期将服务器的负载、cpu利用率、磁盘占用率、服务组件的运行情况等信息发送到中心节点，中心节点根据这些信息作为负载均衡算法的依据。

### 负载均衡策略

在所有的分布式系统中，负载均衡是通过将客户端提交的请求分散到各个服务器节点上从而提高系统的性能。在2.4章节中简单介绍了几种负载均衡的算法策略，一级负载均衡的负载均衡策略采用的是加权法，权值很简单，就是中心节点所对应的应用服务器节点的数量，中心节点所对应的应用服务器节点的数量越多，表示该中心节点所对应的服务器群的并发能力越强，其权值也就越大。中心节点的权值会根据服务器节点数量进行动态的变化。

二级负载均衡采用的IceGrid提供的四种通用的负载均衡策略，随机（random）、自适应（adaptive）、轮询（roundRobin）、优先级（ordered）。应用层的服务管理人员可以通过集群监控模块动态的更改二级负载均衡策略。

## 集群管理监控模块设计

管理监控模块是服务管理人员用来配置、监控OGC应用服务层，主要功能点如下：

1）监控集群内所有应用服务器节点的信息，包括应用服务器节点的IP、内存使用率、磁盘剩余情况、负载信息、服务组件调用信息等。

2）部署应用服务器组件，利用IcePatch2服务与IceStorm服务进行服务组件部署和更新。

3）控制应用服务组件的开启和关闭，通过管理监控模块，可以对集群内的应用服务组件进行单个或者批量的开启或关闭。

4）配置应用服务器组件的负载均衡策略，如4.2章节所述，应用服务组件的第二级负载的四种均衡策略，可以通过管理监控模块进行动态的配置。

在上述各个功能中，集群管理监控模块是直接与各个中心节点进行交互，中心节点与各个应用服务器节点进行交互，如图4.3为利用管理监控模块发布服务的过程：



4.3 管理监控模块进行服务部署

由4.x可知，OGC应用服务组件的发布流程为：首先，服务管理员通过管理监控的终端将新的服务组件和xml配置信息发往各个中心节点；中心节点将新的服务组件根据xml配置信息添加到服务集群中，利用IcePatch2服务奖服务组件发送到各个应用服务器节点；管理监控模块利用中心节点发送命令将新配置的服务组件开启。

## 分布式缓存模块设计

### 概述

在高并发的分布式服务器系统的研究中，分布式缓存是一个热点方向，引入缓存技术，通过在应用服务器节点内部开辟缓存服务或者部署在独立的缓存服务器上，将会大大缓解整个分布式服务的负载压力，提高数据请求的访问效率。

缓存服务器节点主要消耗的是内存资源，而对CPU占用率并不高，而OGC应用服务器对CPU的占用率较高，对内存并没有充分占用。从资源充分利用的角度，在逻辑功能角度，分布式缓存服务器集群和分布式OGC应用服务器集群是两个独立的集群；在物理部署调度，这两个集群公共相同的服务器集群节点。

分布式缓存模块的功能有两点：1）对OGC应用服务器接收的服务请求的结果数据进行缓存，当在一定时间内相同请求再次到来时，可以直接从缓存服务器中获取结果数据，避免OGC应用服务器再次进行逻辑计算处理，提高服务处理效率。2）针对WMS服务的地图瓦片实时渲染请求，对周边数据进行渲染任务，结果放入缓存服务器节点中，根据空间局部性，当周边的渲染请求再次到来时，直接从缓存服务器节点中获取结果数据，从而提高服务效率。

### Memcached缓存服务集群

OGC应用服务器层的缓存模块的设计采用了开源的分布式对象缓存系统Memcached。Memcached缓存服务器集群基于分布式Key-Value模型，每一个缓存服务器节点相互之间不进行通信，主要面向较小的对象，对于WFS的获取要素操作和打包下载服务操作，当其数据结果超过一定的阈值时，若对其进行缓存，会占用缓存服务器节点大量的内存资源，导致其他正常的结果数据无法缓存，所以这一小部分数据拒绝缓存。

Memcached缓存的分布式功能是通过客户端来实现，缓存集群内部每一个节点并没有“分布式”的功能。Memcached客户端作为Memcached缓存集群的中心节点，管理缓存集群，根据一致性hash算法，支持动态得向缓存集群中添加和删除缓存节点。加上缓存服务器之后，Web客户端的请求流程如下4.4：



4.4 异步会话+分布式缓存下的服务请求过程

1）Web客户端根据一级负载均衡选择某中心节点，向其获取OGC应用服务器集群内的某服务器节点上的服务代理信息。

2）Web客户端根据代理信息向应用服务器节点发送请求。

3）应用服务器接收到服务请求之后向缓存服务器中心节点查询该请求任务的数据是否已缓存，若已缓存则直接获取缓存数据返回给客户端，否则进入4

4）应用服务器节点按照3.2章节中所述的基于线程池的异步会话模型处理该请求任务，将结果数据返回给客户端。

5）判断缓存数据的大小是否超过阈值，若没有超过则将数据存入缓存服务器。

### 数据预处理与空间局部性

当用户在使用分布式OGC服务框架平台时，大多数请求的目的是在线浏览地图数据，主要使用了WMTS服务和WMS服务，无论是处理好的地图切片数据还是实时渲染的地图数据的在线浏览都有两个特点：热点数据与空间局部性。

显然，对于一些热门地区的地理数据的浏览量比平均水平要高出很多，对应于这些地区的数据，在一些特殊时期（如“双十一”期间的物流跟踪）会存在大量的并发请求，对服务器产生高并发的访问压力。分布式缓存服务器集群可以大幅度的降低热点数据造成的并发压力，但是缓存服务器发挥作用的前提是缓存中有热点数据，在服务器刚启动的一段时间里，缓存服务器中的数据量非常少，此时无法对OGC应用服务器层起到分担负载的作用。

数据预处理：在OGC应用服务刚启动时，利用客户端主动模拟一些数据数据地区的服务请求，让缓存得到预热。经过数据预处理后的应用服务器层，在热点数据地区的大量并发请求来临时，分布式缓存服务集群已经做好了充分的预热准备，可以为应用服务器层分担并发负载，保持服务的高效和稳定。

空间局部性：用户在利用WMTS或WMS服务进行地图浏览的时候，根据用户的漫游操作习惯会引发局部性数据，当浏览的视口发生漫游移动时，必然会对当前请求的地区的周边地图数据产生访问请求。根据空间局部性的特性，我们可以提前对当前访问请求的周边数据进行处理，待到周边请求过来的时候，此部分的数据已近准备好，这样的提前处理会让前端使用者在使用服务时感觉非常流畅，服务整体的体验有相当大的提升。

根据以上空间局部性的描述，应用服务器在对当前任务的周边数据进行提前处理后，不能将周边结果数据直接存入本机的结果队列中，等待下一次请求将结果拿走。因为对于OGC应用服务器集群的中心节点来说，每一次请求之间都是无关联的，中心节点无法去定位是集群中哪一台应用服务器节点对上一次任务请求做了周边数据预处理，当下一次周边数据请求到来的时候，中心节点仍然是根据二级负载均衡策略返回某应用服务器的代理信息，如此一来，预处理的数据就无法被客户端的服务请求获取到，还造成了应用服务器的资源的浪费。所以不能将周边的数据结果存放在某特定的应用服务器上，我们采取的方案是将周边数据的结果放入分布式缓存服务器中，流程图如下4.5：



4.5 空间局部性处理流程

1）Web客户端根据一级负载均衡策略和二级负载均衡策略定位到某特定的应用服务器节点派发数据请求任务。

2）应用服务器节点接收到数据请求之后将该任务添加到线程池开始处理，同时生成周边数据的请求任务。

3）如3.2章节所述，Web客户端和OGC应用服务器之间基于线程池的异步会话模型将此次请求结果数据进行传输，而OGC应用服务器的预处理数据结果则主动发送到缓存服务器集群。

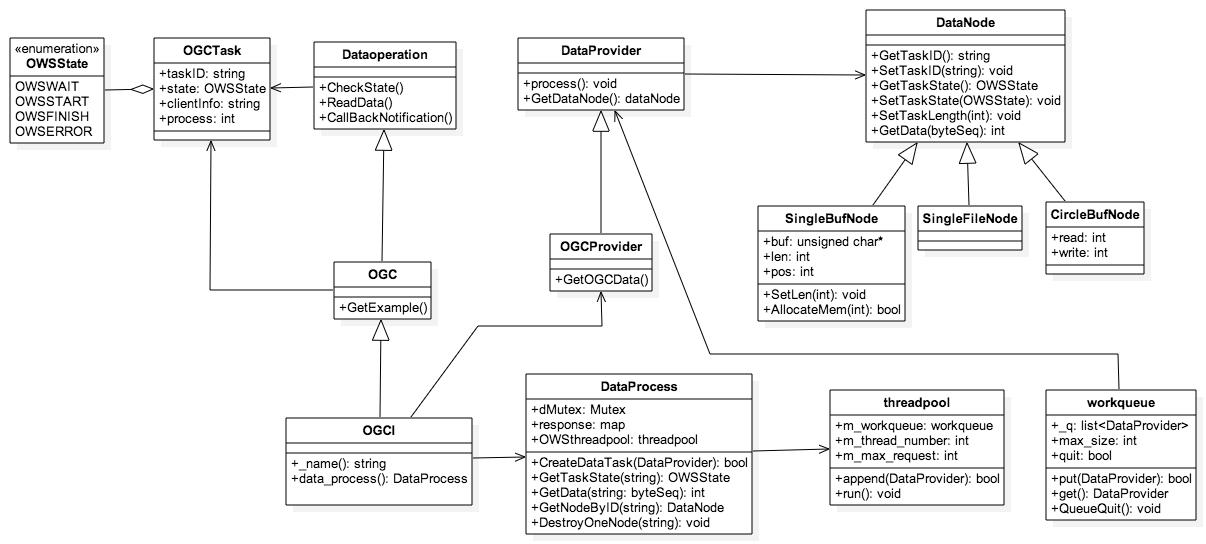
4）当Web客户端的周边数据请求再次定位到某应用服务器节点上时，便直接从缓存服务器集群中将数据获取并返回给客户端程序。

# OGC应用服务器层的实现与测试

## OGC应用服务器集群实现

### OGC应用服务组件开发

由OGC应用应用服务器设计可知，OGC应用服务器组件开发最终采用了异步会话模型。OGC应用服务器为每一个客户端请求返回一个OGCTask结构体，该结构体的数据包含了请求任务唯一标识符taskID（实际是全球唯一标识符uuid）、请求任务的状态（等待、可读、完成、错误）、任务完成进度、客户端回调的代理信息。Web客户端既可以通过该OGCTask结构体向服务器轮询请求的状态，应用服务器端也可以根据任务体内的回调代理信息显示通知Web客户端任务状态，当请求任务完成后Web客户端再向应用服务器获取请求结果。

OGC应用服务器组件共实现了WFS、WMS、WCS、WMTS、BasicInfo、PackageDownload六种服务组件。每种服务组件的功能不同，但流程框架都是相同的，我们抽象出一种OGC服务组件来说明整个OGC应用服务器组件的实现，类图如下5.1: 

5.1 OGC整体服务类图

在5.1OGC整体服务类图中，在实现新的OGC服务组件时，通过slice接口描述语言定义组件内容，并通过slice2cpp生成组件的接口类，例如OGC类和其服务方法GetExample，应用服务器端开发者通过继承OGC类实现OGCI类完成方法的具体实现。OGCProvider类继承了DataProvider类，实现了服务组件中数据获取的方法。结果数据继承了DataNode体系，按照不同服务组件的特性，结果数据可以存放在单一内存块中（SingleBufNode），也可以落盘放入文件中（SingleFileNode，如打包下载服务），也可以利用循环内存缓冲区（CircleBufNode，WFS的GetFeatrue操作），按照新的服务组件的需求，可以动态扩展其他形式的存储方法。

任务处理工厂DataProcess中实现了应用服务器端的线程池和任务队列，开发者只需将特定服务组件的数据获取类的实例交给DataProcess，线程池即可按照优先级策略去调用数据获取方法，并将结果放入结果队列中，最后更新OGCTask中的任务的处理状态（由OWSWAIT更新为OWSSTART）。在任务处理完毕之后，应用服务器端通过调用CallBackNotification方法通知Web客户端可以调用ReadData读取结果数据。如果OGC应用服务器端处理请求的时间较长，Web客户端可以通过CheckState方法获取最新的OGCTask结构体信息，查询任务处理的进度信息。

### OGC应用服务器集群两级负载均衡的实现

根据4.2章节负载均衡的模型设计和负载均衡策略，

## 集群监控的实现

## 分布式缓存的实现

# 总结与展望

参考文献

1. 普通图书示例：
2. 学位论文示例：
3. 张志祥. 间断动力系统的随机扰动及其在守恒律方程中的应用[D]. 北京: 北京大学数学学院, 1998.
4. CALMS R B. Infrared spectroscopic studies on solid oxygen[D]. Berkeley: Univ. of California. 1965.
5. 专利文献示例：
6. 刘加林. 多功能一次性压舌板: 中国, 92214985. 2[P]. 1993-04-14.
7. 河北绿洲生态环境科技有限公司.一种荒漠化地区生态植被综合培育种植方法: 中国, 01129210.5[P/OL]. 2001-10-24[2002-05-28]. http://211.152.9.47/sipoasp/zlijs/hyjs-yx- new.asp? recid=01129210.5&leixin.
8. KOSEKI A, MOMOSE H, KAWAHITO M, et al. Compiler: US, 828402[P/OL]. 2002-05-25 [2002-05-28].http://FF&p=1&u=netahtml/PTO/search-bool.html&r=5&f=G&1=50&co1=AND&d=PGOl&sl=IBM. AS. &OS=AN/IBM&RS=AN/IBM.
9. 专著中析出的文献示例：
10. 国家标准局信息分类编码研究所. GB/T 2659-1986世界各国和地区名称代码[S]//全国文献工作标准化技术委员会. 文献工作国家标准汇编: 3.北京: 中国标准出版社, 1988:59-92.
11. 韩吉人. 论职工教育的特点[G]//中国职工教育研究会. 职工教育研究论文集. 北京: 人民教育出版社, 1985:90-99.
12. BUSECK P R, NORD G L, Jr. , VEBLEN D R. Subsolidus phenomena in pyroxenes[M]// PREWITT C T. Pyroxense. Washington, D. C. : Mineralogical Society of America, c1980: 117-211.
13. FOURNEY M E. Advances in holographic photoelasticity [C]// American Society of Mechanical Engineers. Applied Mechanics Division. Symposium on Applications of Holography in Mechanics, August 23-25, 1971, University of Southern California, Los Angeles, California. New York: ASME, c1971 : 17-38.
14. MARTIN G.. Control of electronic resources inAustralia[M]//PATTLE L W, COX BJ. Electronic resources: selection and bibliographic control. New York: The Haworth Press, 1996: 85-96.
15. 期刊中析出的文献
16. 李炳穆. 理想的图书馆员和信息专家的素质与形象[J]. 图书情报工作, 2000(2):5-8.
17. 陶仁骥. 密码学与数学[J]. 自然杂志, 1984,7(7):527.
18. 习亚洲地质图编目组. 亚洲地层与地质历史概述[J]. 地质学报, 1978, 3: 194-208.
19. DES MARAIS D J, STRAUSS H, SUMMONS R E, et al. Carbon isotope evidence for the stepwise oxidation of the Proterozoic environment [J]. Nature, 1992, 359: 605-609.
20. HEWITT J A. Technical services in 1983[J]. Library Resource Services, 1984, 28(3): 205-218.
21. 报纸中析出的文献示例：
22. 丁文祥. 数字革命与竞争国际化[N]. 中国青年报, 2000-11-20(15).
23. 张田勤. 罪犯DNA库与生命伦理学计划[N]. 大众科技报, 2000-11-12(7).
24. 电子文献(包括专著或连续出版物中析出的电子文献)示例：
25. 江向东. 互联网环境下的信息处理与图书管理系统解决方案[J/OL]. 情报学报, 1999, 18(2): 4[2000-01-18].
26. 萧钮. 出版业信息化迈人快车道[EB/OL]. (2001-12-19) [2002-04-15]
27. CHRISTINE M. Plant physiology: plant biology in the Genome Era[J/OL]. Science, 1998, 281:331-332[1998-09-23].
28. METCALF S W. The Tort Hall air emission study[C/OL]//The International Congress on Hazardous Waste, Atlanta Marriott Marquis Hotel, Atlanta,Georgia, June 5-8, 1995: impact on human and ecological health[1998-09-22].
29. TURCOTTE D L. Fractals and chaos in geology and geophysics[M/OL]. New York: Cambridge University Press, 1992[1998-09-231.
30. Scitor Corporation. Project scheduler[CP/DK]. Sunnyvale, Calif. : Scitor Corporation, c1983.

致谢

本论文是在导师的悉心指导下完成的，从论文的选题到论文的撰写，无不渗透着导师的心血，……值此论文完稿之际，谨对导师的辛勤培育以及谆谆教诲表示最衷心的感谢!

作者简介

##### 基本情况

豆超，男，陕西西安人，1982年8月出生，西安电子科技大学XX学院XX专业2008级硕士研究生。

##### 教育背景

2001.08～2005.07西安电子科技大学，本科，专业：电子信息工程

2008.08～ 西安电子科技大学，硕士研究生，专业：电磁场与微波技术

##### 攻读硕士学位期间的研究成果

###### 发表学术论文

1. XXX, XXX, XXX. Rapid development technique for drip irrigation emitters[J]. RP Journal,UK.,2003,9(2): 104-110.(SCI: 672CZ, EI: 03187452127)
2. XXX, XXX, XXX. 基于快速成型制造的滴管快速制造技术研究[J]. 西安交通大学学报, 2001, 15(9): 935-939. (EI: 02226959521)

###### 申请（授权）专利

1. XXX, XXX, XXX等. 专利名称: 国别,专利号[P]. 出版日期.

###### 参与科研项目及获奖

1. XXX项目, 项目名称, 起止时间, 完成情况, 作者贡献。
2. XXX, XXX, XXX等. 科研项目名称. 陕西省科技进步三等奖, 获奖日期.