|  |
| --- |
| **二、研究目的和意义简述** |
| 随着图书馆各种服务的增多，图书馆机房中实体服务器及虚拟服务器的数量也不断增加，并且由于服务的不同，每台服务器上运行的程序和服务也各不一样，所以仅靠人工方式来确认各个服务器上运行的服务是否正常，不仅加重了管理人员的负担，而且经常不能及时的发现和排除故障，给图书馆的读者服务造成负面影响。  因此，本项目的目的是编写一个服务器检测程序，利用程序来定时对各个服务器上运行的服务状态进行监测，记录各服务器的运行状态，从而可以及时发现服务器的故障并通知相关人员，同时根据预设的规则来评估造成服务故障的原因，并提供建议的解决方案供管理人员参考。这样不仅可以提高系统可靠性，还可减轻服务器管理人员的负担、完善图书馆机房服务器的科学管理，为图书馆各种服务的正常运行提供保障。 |

|  |
| --- |
| **三、主要研究内容** |
| 本项目是在中国海洋大学图书馆前两个版本的服务器监测程序的基础上大幅改进而成。2010年完成的第一版基于Ribbon UI 的服务器检测程序，可以检测服务器的状态及机房环境温度等信息。    在2013年完成的第二版基于Metro UI的服务器检测程序，使用了Entity Framework替换SQL Client，并加入了故障短信提醒等信息。    但第二版的服务器状态监测主要局限于页面关键词抓取，iis、ftp端口检测等方面，无法对监测的服务器进行灵活的配置，也无法获取整个服务器的运行状态信息。  在本项目中，主要使用的是通用的SNMP协议，SNMP协议即简单网络管理协议，由一组网络管理的标准组成，包含一个应用层协议（application layer protocol）、数据库模型（database schema）和一组资源对象。该协议能够支持网络管理系统，用以监测连接到网络上的设备是否有任何引起管理上关注的情况。该协议是互联网工程工作小组（IETF，Internet Engineering Task Force）定义的Internet协议簇的一部分。[1] 与上两个版本只能在外部查看服务器状态不同，本项目通过SNMP协议可以获取服务器的CPU、内存、存储、进程及服务的运行状况等个方面的信息，从总体上评估服务器当前的使用情况，在服务器异常时向 管理员预警，并根据特征库来对管理员提供对服务器维护的建议。  本项目研发的服务器监测系统主要包含数据采集、数据存储、WEB发布、API发布四个模块，系统组成见下图： |

|  |
| --- |
| **四、分年度主要研究任务和实施方案** |
| 在Windows Server的控制面板中选择“打开或关闭 Windows 功能”，来启用服务器的SNMP 服务。启用服务后，在“服务”中可以看到 SNMP 服务已经启动，接着需要更改 SNMP 服务的属性，来限制SNMP服务的权限及可被哪些IP的设备读取。 |

|  |
| --- |
| **五、绩效目标及年度目标** |
| 3. 通过SNMP协议获取服务器相关信息：  1) CPU信息  在Windows服务器中，通过WALK方法读取OID:.1.3.6.1.2.1.25.3.3.1.2，可以获得该服务器各个CPU在最近一分钟内的CPU平均使用率，使用率的数值为整数，获取的值的数量和服务器CPU的核心数相等。通过此OID获取的值，可以计算出服务器在近一分钟内的CPU单核最高使用率及CPU的平均使用率，从而获取服务器CPU的当前使用情况。（获取情况如下图）    2) 存储信息  在Windows服务器中，通过WALK方法读取OID: .1.3.6.1.2.1.25.2.3.1.3，可以获得服务器的存储相关信息，我们可以根据此信息来判断不同存储的类型（如内存、硬盘、光盘等）。如下图：    了解了存储的分类信息，我们可以继续使用WALK方法读取OID: .1.3.6.1.2.1.25.2.3.1.4，可以获得每个存储的簇的大小。（如下图）    继续使用WALK方法读取OID: .1.3.6.1.2.1.25.2.3.1.5，可以获得每个存储的簇的数目。（如下图）    继续使用WALK方法读取OID: .1.3.6.1.2.1.25.2.3.1.6，可以获得每个存储已经使用的簇的数目。（如下图）    得到了这些信息，比如我们要计算服务器上C盘的存储信息，可以计算：  C盘已用容量：  4096（簇的大小） \* 26188287（已使用簇的数量）= 50848079872 bytes ≈ 47.36G  C盘全部容量  4096（簇的大小） \* 12414082（簇的数量）= 107267223552 bytes ≈ 99.90G  有了硬盘的容量，我们可以计算C盘的使用率为：47.4%  同样的方法，我们也可以计算服务器的内存使用情况，比如此服务器的内存已使用容量为：  65536（簇的大小） \* 33346（已使用簇的数量）= 2185363456 bytes ≈ 2.04G  内存全部容量  65536（簇的大小） \* 131064（已使用簇的数量）= 8589410304 bytes ≈ 8.00G  计算内存的使用率为：25.5%  通过以上方法，我们可以对应的获取服务器的存储及内存的使用情况。 |

|  |  |
| --- | --- |
| **六、总体考核指标（具体、量化）** | |
| 解决的关键问题或技术难点 |  |
| 主要技术指标 |  |
| 预期成果 |  |
| 推广应用及经济社会效益 |  |
| 人才队伍建设 |  |
| 适应推广的技术模式 |  |
| 其他 |  |

备注：可根据实际情况调整考核指标类型。

|  |
| --- |
| **七、工作基础** |
| 通过SNMP协议读取OID信息  SNMP协议通过读取对应OID来获取对应的信息。OID（Object Identifier，对象标识符又称为物联网域名）是由ISO/IEC、ITU-T国际标准化组织上世纪80年代联合提出的标识机制，采用分层树形结构对任何类型的对象（包括实体对象、虚拟对象、复合对象等）进行全球无歧义、唯一命名。OID具有分层灵活、扩展性强、跨异构系统等优势，并可兼容现有标识机制，已经广泛应用于信息安全、医疗卫生、网络管理等领域。[2]  SNMP读取OID信息，有GET和WALK两种方式：GET为获取当前节点的值，WALK为获取当前节点所有子节点的值。  1) GET方式获取SNMP的值：  /// <summary>  /// GET方法获取SNMP信息  /// </summary>  /// <param name="host">服务器名或IP</param>  /// <param name="key">SNMP口令</param>  /// <param name="oid">OID</param>  /// <returns></returns>  private string GetOidData(string getoidname, string host, string key, string oid)  {  string str = string.Empty;  try  {  OctetString community = new OctetString(key);  // Define agent parameters class  AgentParameters param = new AgentParameters(community);  // Set SNMP version to 1 (or 2)  param.Version = SnmpVersion.Ver1;  // Construct the agent address object  // IpAddress class is easy to use here because  // it will try to resolve constructor parameter if it doesn't  // parse to an IP address  IpAddress agent = new IpAddress(host);  // Construct target  UdpTarget target = new UdpTarget((IPAddress)agent, 161, 2000, 1);  // Pdu class used for all requests  Pdu pdu = new Pdu(PduType.Get);  pdu.VbList.Add(oid); //sysName  // Make SNMP request  SnmpV1Packet result = (SnmpV1Packet)target.Request(pdu, param);  // If result is null then agent didn't reply or we couldn't parse the reply.  if (result != null)  {  // ErrorStatus other then 0 is an error returned by  // the Agent - see SnmpConstants for error definitions  if (result.Pdu.ErrorStatus != 0)  {  // agent reported an error with the request  log.Error(host + " " + (string.Format("Error in SNMP reply. Error {0} index {1}",result.Pdu.ErrorStatus,result.Pdu.ErrorIndex)));  }  else  {  // Reply variables are returned in the same order as they were added  str = result.Pdu.VbList[0].Value.ToString().Trim();  }  }  else  {  log.Error(host + " ,No response received from SNMP agent.");  }  target.Close();  }  catch (Exception ex)  {  log.Error(ex.Message);  str = "";  }  return str;  }  2) 通过WALK方式获取SNMP的值  /// <summary>  /// WALK方法获取SNMP信息  /// </summary>  /// <param name="walkoidname"></param>  /// <param name="host">服务器名或IP</param>  /// <param name="key">SNMP口令</param>  /// <param name="oid">OID</param>  /// <returns></returns>  private List<SnmpResult> GetOidList(string walkoidname, string host, string key, string oid)  {  List<SnmpResult> slist = new List<SnmpResult>();  try  {  SimpleSnmp snmp = new SimpleSnmp(host, key);  if (!snmp.Valid)  {  log.Error(host + " , " + walkoidname + " ,SNMP agent host name/ip address is invalid.");  slist.Clear();  return slist;  }  Dictionary<Oid, AsnType> result = snmp.Walk(SnmpVersion.Ver2, oid);  if (result == null)  {  log.Error(host + " , " + walkoidname + " ,No results received.");  slist.Clear();  return slist;  }  if (result.Count > 0)  {  foreach (KeyValuePair<Oid, AsnType> kvp in result)  {  slist.Add(new SnmpResult()  {  Oid = kvp.Key.ToString(),  Type = SnmpConstants.GetTypeName(kvp.Value.Type),  Value = kvp.Value.ToString().Trim()  });  }  }  else  {  log.Error(host + ",Results Count is zero.");  slist.Clear();  return slist;  }  }  catch (Exception ex)  {  log.Error(ex.Message);  slist.Clear();  }  return slist;  } |