匹配场声源定位的并行计算方法研究 *

董姝敏[1,2], 刘洪波[1], 赵 博[2,3], 梁国龙[2]
(1. 吉林师范大学 信息技术学院, 吉林 四平 136000; 2. 哈尔滨工程大学 水声技术国防科技重点实验室, 哈尔滨 150001; 3. 海军 92330 部队, 山东 青岛 266061)

摘 要: 为了达到有效缩短算法运行时间、加快定位速度的目的,研究并分析了匹配场声源定位算法中存在的固有并行性,构建了匹配场声源定位的并行计算算法框架。详细介绍了匹配场声源定位算法的网格法并行化方法;并以互连的双核 PC 机作为并行计算硬件平台,选用 Windows 操作系统、FORTRAN 编译器,使用 TCP/IP 作为标准的通信协议,在 MPI 并行环境中对并行算法进行了测试和相应分析。研究表明,提出的并行算法运行效果较好,可以通过开发匹配场处理方法的并行性,有效解决信号匹配处理中遇到的计算量大、耗时长等问题,为实现快速声源定位提供解决途径。

关键词: 匹配场处理; 声源定位; 并行处理; MPI 并行环境
中图分类号: TP31 **文献标志码**: A **文章编号**: 1001-3695(2012)02-0514-04
doi:10.3969/j.issn.1001-3695.2012.02.030

Study on method of parallel computing in match field localization

DONG Shu-min[1,2], LIU Hong-bo[1], ZHAO Bo[2,3], LIANG Guo-long[2]
(1. *College of Information Technology, Jilin Normal University, Siping Jilin* 136000, *China*; 2. *National Laboratory of Underwater Acoustic Technology, Harbin Engineering University, Harbin* 150001, *China*; 3. *The* 92330 *Force of Navy, Qingdao Shandong* 266061, *China*)

Abstract: Through studying and analyzing the inherent parallelism which existed in algorithm of the match field sound source localization, frame of parallel computing algorithm had constructed in the match field localization in order to shorten the running time and accelerate the speed of localization algorithm effectively. Parallelization grid method was introduced in detail , which was applied to the match field localization algorithm. And parallel algorithm had been carried on the test and the corresponding analysis in the MPI parallel environment, in the way of interconnecting binuclear PC machines as the parallel computing hardware platform through fast internet, choosing Windows operate system and FORTRAN compiler, and using TCP/IP as the standard communication protocols. The study shows the proposed parallel algorithm running results are better,so questions of the load big computation and long time-consuming etc. can be settled effectively to exploiting parallelization in matched field processing methods, in order to provide settle path for fast sound source localization.

Key words: matched field processing; sound source localization; parallel processing; MPI parallel environment

0 引言

匹配场处理技术已经在水声匹配场参数反演、匹配场声源定位、匹配场噪声抑制等方面得到广泛关注。其中,匹配场声源定位方法能够突破传统被动定位方法的极限,正确估计远程和超远程声源的距离和深度。但由于匹配场定位技术需要利用声传播模型来反复计算拷贝场向量,然后将拷贝场与测量场进行“匹配”,进而实现水下目标的定位;另外,在实际应用中,由于水声环境条件十分复杂,并且存在较大的时空变化,匹配场定位方法面临着计算拷贝场的计算量大、耗时长、占用存储空间大[1]等问题,难以在允许的时间内为仿真环境提供令人满意的数据支持,因此提高算法的计算速度显得特别重要,尤其是随着问题规模的扩大,提高速度已经势在必行。并行计算[2]能够将处理任务合理地分配到计算机系统的多个处理机上,使各处理机的工作负载保持相对均衡,整个计算机系统在较短的时间内协同完成处理任务,从而加快计算速度,提高计算效率。因此利用匹配场声源定位算法中存在的固有并行性,采用并行处理技术来解决匹配场定位中的计算量大的难题具有重要的研究意义和实际应用价值,而目前国内外关于匹配场定位的并行计算的文献并不多见。本文在 KARKEN 计算软件的基础上实现匹配场定位的并行计算,在保证一定精度的前提下有效缩短程序运算时间。

1 匹配场定位中并行性分析

声场建模、拷贝场计算、相关处理是匹配场声源定位的重要研究内容。声场模型和拷贝场的计算精度和速度及匹配场处理算法的选取等决定了匹配场声源定位的精度与速度。首先,获取实际测量声场(简称测量场)数据,它是涵盖了信道特

收稿日期: 2011-08-02; **修回日期**: 2011-09-12 **基金项目**: 国家自然科学基金资助项目(51009042); 高等学校博士学科点专项科研基金资助项目(20102304120030); 黑龙江省自然科学基金资助项目(E201024)

作者简介: 董姝敏(1978-),女,吉林松源人,博士,主要研究方向为盲信号处理、智能计算(dongsm372@126.com);刘洪波,男,讲师,硕士,主要研究方向为光电信息处理、数字图像处理;赵博,男,硕士,主要研究方向为并行计算、声场计算;梁国龙(1964-),男,教授,博导,主要研究方向为水声工程、水声信号信号处理.

征和声源特征的声压场;其次,根据声场模型与已知的环境参数(如声速分布、海底地形等),对假定的声源进行声场(简称参考场)计算;最后,利用匹配场处理算法对实验测得的声场(测量场)与建模获得的声场(拷贝场)作相关处理,找到与测量场匹配得最佳的参考场,则计算该参考场时所假定的声源位置就认为是实际的声源位置[3],以此估计声源的距离和深度,从而实现对水下目标的定位,如图 1 所示[4]。

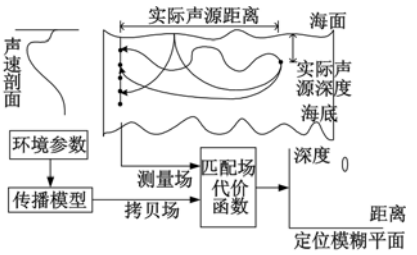

图1　匹配场定位原理示意图

1.1　简正波解的并行性

选择简正波理论进行拷贝声场计算,主要是因为简正波理论适合于水平分层介质,相对于其他模型,计算简单、速度快,只需求一次本征值即可反复计算声场,可方便计算整个声场。但一般认为简正波理论适用于低频、远场、水平分层介质情况,因为声波的频率较高或海深较深时,简正波阶数较多,计算量增大;宽带信号或脉冲信号的求解,只能逐个频率单独计算再叠加合成,计算量很大;非水平分层介质问题,对海底地形、声速、密度等海洋环境参数变化剧烈的传播问题只能用耦合简正波方法,计算量非常巨大。限制简正波理论在实际应用中的主要瓶颈是计算量大,若能采用并行计算技术使各并行处理机分别同时求解各阶简正波的解,必将提高简正波的求解速度,减少计算时间,简正波理论的实用限制条件就可以宽松得多。目前基于简正波理论的声场计算软件有 KARKEN、SNAP、NSAPRD 和 ORCA 等。KARKEN 软件使用理想情况下的简化方法求解简正波的特征值和特征函数[5],各阶简正波之间并无十分紧密的联系,即可以看成是相互独立的,参加并行计算的每台处理机分别负责某些阶数简正波特征值和特征函数的计算,实现并行化。文献[6]已详细介绍了 WKBZ 简正波解的并行化计算问题。

1.2　拷贝场计算中的并行性

拷贝场的计算方法对于提高声源定位的精度和计算速度至关重要。匹配场声源定位的主要计算量来自拷贝场计算,需要在计算所覆盖的区域内进行距离/深度空间采样,每个采样点处假定存在一个声源,假定声源位于网格节点上,声源的距离和深度可当做已知量,网格点数由信号频率已拷贝场区域范围确定。然后采用简正波方法的 KRAKEN 软件计算假定声源的声压,以此得到拷贝场。

可见拷贝场的计算需要获得数以千计甚至是数以万计个假定声源的声压场,各声压场的计算是相互独立的,可以方便地实现并行计算,每台处理机分别单独地负责某些网格点的声压场计算即可,而且无须通信,可有效缩短拷贝场计算时间。

1.3　相关处理的并行性

用匹配场处理算法对测量场与参考场作匹配相关处理时,需要分别对各个假定声源与实际声源作相关,相关次数与假定声源的数目相同,而且各次相关处理之间互不影响,将相关处理的任务合理地分配到计算机系统的多个处理机上,整个计算机系统在较短的时间内完成相关处理。

匹配场定位中也存在其他的并行机制,如进行匹配处理过程中还可以实现功能级并行处理。本文只针对计算量较大占用时间相对较长的拷贝场计算及相关处理实施并行化处理。

2　并行计算系统结构

MPI(message passing interface)是国际上通用的消息传递接口,是并行程序设计标准之一,它可与 C、FORTRAN、C++、Java 等语言结合而构成并行程序设计语言。MPI 有许多实现版本,其中 MPICH 提供了接口一致的 MPI 标准函数库和程序运行环境。另外,MPI 不仅支持各种硬件环境,几乎支持各种操作系统 Linux、UNIX 和 Windows 等。因此,采用 MPI 编写的并行程序可移植性好,普通的 PC 机加上一个高速局域网即可实现,价格便宜,易于实现,并且有成熟的软件开发工具,得到了广泛应用。

2.1　软、硬件系统配置

采用基于 Windows 的 PC 机群系统实现并行计算。系统的具体配置情况如下:

a)硬件组成。双核 PC 机(Intel PentiumⅣ)、网络交换器。

b)软件组成。Windows XP 操作系统、MPI 并行环境、Microsoft. NET Framework 2.0、FORTRAN 编译器。

2.2　MPI 并行环境搭建

a)每台 PC 机中新建一个 MPI 用户,该用户应该具有管理员权限,隶属 Administrators 组,各台 PC 中的 MPI 用户名和密码均相同。

b)在各台 PC 机上均安装 Microsoft. NET Framework 2.0 和 MPICH2,各台 PC 机安装 MPICH2 时口令要一致,默认口令是“be happy”。MPICH2 的默认安装路径为 C:\Program Files\MPICH2。

c)运行 MPICH2 中的 wmpiregister. exe 注册用户。

d)用网络交换器将 PC 机连接,把待测试的可执行程序拷贝到每台 PC 机的同一目录中,该目录应该在相同的位置,如 C:\MPI Program 下,然后将 Windows 系统的防火墙关闭。

e)用 MPICH2 自带的界面方式运行可执行程序,选择“more option”打开下拉对话框,使用 IP 地址或者 PC 机器名连接各台 PC 机,空格分隔。

3　匹配处理算法(相关处理)

自适应匹配场处理器将自适应阵列信号处理的优点融合到匹配场处理中,能够有效抑制旁瓣和干扰,提供了理论上的最佳阵增益和定位精度,比线性处理器具有更好的性能[7]。本文选用使测量场数据与拷贝场数据在输出噪声功率最小的意义上构建的最小方差处理器。最小方差处理器处理器的代价函数计算公式[8]为

$$P_{mv}=\frac{1}{\hat{F}^{+}\overline{\hat{C}}^{-1}\hat{F}}$$

其中:$C=\langle FF^{+}\rangle$,$F=(F_1,F_2,\cdots,F_N)^{\mathrm{T}}$;$N\times N$,$F_n$ 代表第 n 个水听器的测量声场,$\|F\|=\sqrt{F_1^2+\cdots+F_N^2}=1$;$\hat{F}_n$ 代表第 n 个水听器的拷贝声场,$\|\hat{F}\|=\sqrt{\hat{F}_1^2+\cdots+\hat{F}_N^2}=1$;$F^{+}$ 代表 F 的共轭转置;F^{T} 代表 F 的转置;N 代表水听器的个数;C 是 $N\times N$

的互谱矩阵。

4　拷贝场计算及相关处理并行实现

并行计算的算法实现过程如图 2 所示。进程 0 计算测量场声压,使用函数 MPI_bcast( )将该声压广播给其他进程;所有进程同步计算负责区域的拷贝场声压,之后将拷贝场声压与测量场声压进行相关处理,直到所有进程都处理完毕,各进程的同步由函数 MPI_barrier( )实现;由某个指定进程负责收集各个进程的处理结果,由 MPI 函数 MPI_gather( )实现,最后打印输出统计处理结果。

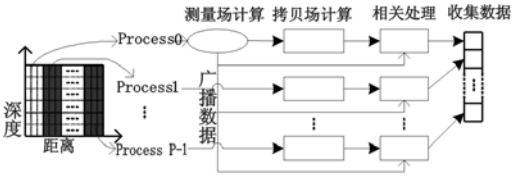

图2　匹配场并行化计算框图

拷贝场计算是匹配场处理中最耗时的部分,为了获得较高的定位精度,必须将声场的网格划分得很小。因此程序运行的时间将会很长,即使进行并行计算,如果并行运行的进程数较少,其运行时间也需要十几至几十分钟,根本无法满足软件实时性的要求,因此要折中考虑并行计算的进程数和划分的网格数的选取问题。

4.1　任务划分

设有 P 个进程(进程序号 id = 0,1,…,$P-1$)参与并行计算,并将拷贝场覆盖区域划分为 $M \times N$ 点网格,即在深度方向上 M 点采样、距离方向上 N 点采样。整个拷贝场计算和相关处理作为总任务,每个网格的声压计算和匹配相关处理作为一个子任务,深度方向上的 M 个子任务作为一个子任务块,这样距离方向上就有 N 个子任务块,任务块序号依次为 1,2,…,$N-1$,N。

拷贝场中各个网格点声压 P_2 的计算是独立的,无须通信,因此各个进程分别负责部分区域的各点 P_2 的求解;P_2 与测量场声压 P_1 的相关处理也独立于其他点的处理,只需把处理的结果传递给某个进程统一处理。

4.2　分配总任务

分配总任务主要有交叉分配和按块分配两种方法[9]。

交叉分配就是将 N 个子任务块依次分配给 P 个进程负责,各进程负责的子任务块序号分别为

进程 0 负责 1,1 + P,1 + 2P,…

进程 1 负责 2,2 + P,2 + 2P,…

…

进程 P-1 负责 P,2P,3P,…

交叉分配的好处是很容易确定某个子任务块 i 由哪个进程负责,即由进程 mod(i,P)负责,这种分配任务方式用于动态分配模式较适合。

按块分配意味着将 N 个子任务块分为 P 个连续的块,每块的大小基本相等,一个进程负责一块。如果 N 不是 P 的倍数,需要考虑负载平衡问题,给每个进程分配⌈N/P⌉或⌊N/P⌋个子任务。计算 R = mod(N,P)。如果 R = 0,每个进程分配 N/P 个子任务;如果 75%,第一种方案将较大的块分配给前面的进程,即前 R 个进程分配⌈N/P⌉个子任务,其余进程分配⌊N/P⌋个子任务。进程 id 控制的第一个子任务块序号为 low = 1 + id × ⌊N/P⌋ + min(R,id),最后一个子任务块序号为进程 id + 1 控制的第一个序号的前一个子任务块,即 high = (id + 1) × ⌊N/P⌋ + min(R,id + 1),对于特定的子任务块 j 由进程 min(⌊(j - 1)/(⌊N/P⌋ + 1)⌋,⌊j - R)/⌊N/P⌋⌋)控制。

第二种方案将较大的块交替均匀地分配给各个进程,即⌈N/P⌉,⌊N/P⌋,⌈N/P⌉,⌊N/P⌋,…。进程 id 控制的第一个子任务块序号为 low = 1 + ⌊id × N/P⌋,最后一个子任务块序号为进程 id + 1 控制的第一个序号的前一个子任务块,即 high = ⌊(id + 1) × N/P⌋,对于特定的子任务块 j 由进程⌊($P \times j$ - 1)/N⌋控制。

4.3　数据通信

a)进程 0 计算完毕测量场声压后,需要将 P_1 广播给其他进程,其他进程得到 P_1 后才能进行匹配处理,通过调用 MPI 库函数 MPI_bcast( )实现。

b)进程 0(也可是其他指定进程)收集所有其他进程匹配处理后的计算结果,调用 MPI 库函数 MPI_gather( )实现。

c)进程 0(也可是其他指定进程)对收集的结果进行统计处理后,输出打印到相关应用平台上。

5　仿真实验

匹配场处理的距离取值范围为 0.0590.0 km,深度取值范围为 10.0250.0 m,将声场划分为水平间距为 0.05 km、垂直间距为 10.0 m 的网格,参考声源位于网格交叉点上。实际声源距离为 45 km,深度为 100 m,频率 10 Hz。接收基阵由三个基元组成的水平阵(垂直阵),基元间距为 14.84 m,基阵与深度方向夹角 $\theta = 90^\circ$,声源方位角 $\varphi = 0^\circ$,中心基元深度 z_0 = 100 m。

4 台双核 PC 机通过网络交换器互连,安装 Windows XP 操作系统,在 MPICH 并行环境中对并行算法进行测试。对拷贝场计算和相关处理同时进行并行处理的运行结果如表 1 所示。

表 1　参数指标及性能评价表

进程数	运行时间/s	加速比	并行效率/%
1	712.362 4	-	-
2	421.572 6	1.689 8	84.488 7
3	313.132 7	2.275 0	75.831 8
4	257.658 2	2.764 8	69.118 9
5	218.247 7	3.264 0	65.280 2
6	195.588 7	3.642 1	60.702 4
7	176.578 2	4.034 3	57.632 3
8	164.785 7	4.323 0	54.037 0

并行加速比可简单定义为:最好的顺序算法的执行时间与并行算法执行时间的比值,即单进程的执行时间与多进程执行时间的比值[10]。并行效率表示多处理器系统用于进行有用计算的时间百分比,表 1 中并行效率的数据表明多个进程执行时,随着进程数的增加用于进程间通信的开销相对计算来说比例增加。加速比随进程数变化情况如图 3 中的 * 线所示,□线为理想情况加速比。影响加速比未达到最大值 P 的主要因素来自两方面:a)由于程序中的有些部分不能分成并行进程来执行,只能单进程执行,此时其他进程处于闲置状态,随着进程数增加,串行部分所占比例增加;b)进程数增加,进程间通信开销的比例相对增加。当然其中也必然存在计算机软硬件系统的偶然因素,分析起来比较复杂,这里不再讨论。

图 4 是各种并行情况下实施并行计算所获得的加速比性

能对比曲线,斜率从大到小分别指的是:理想情况加速比、拷贝场计算和相关处理同时并行的加速比、仅对拷贝场计算并行化的加速比、仅对相关处理并行化的加速比以及仅并行求解简正波解的加速比。可见,简正波解计算量相对于总的匹配场处理的计算量来说,影响甚微;相关处理次之;拷贝场计算的计算量是匹配场计算量的主要来源,对其进行并行处理,可大大节省处理时间。

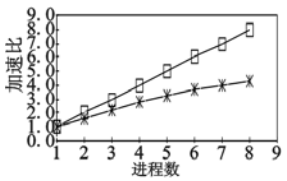

图3　加速比性能曲线

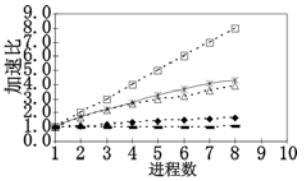

图4　加速比性能对比曲线

通过实验和分析表明,采用并行计算技术进行匹配场定位可以有效节省计算时间,从而提高计算效率。

6　结束语

匹配场声源定位中,需要进行大量重复计算,将整个计算问题划分成若干子部分,使用具有多个计算内核的多核计算机和多个互连计算机作为计算平台,每个处理器并行地完成某个子部分的计算,这样可以突破单台计算机计算性能有限的局限,在实际工程计算中具有广阔的应用前景。本文通过对多核计算机以及计算机机群系统的并行技术进行研究,利用 MPI 消息传递接口编写并行应用程序,应用对等模型;并使用 MPI 标准测试函数,在两个互连的双核计算机上,以简正波理论的声场计算软件 KARKEN 为基础,对匹配场声源定位进行了测试和相应的分析,构建了匹配场声源定位的并行计算算法框架。仿真实验表明:采用并行化技术实现匹配场声源定位,可有效降低计算量,缩短计算时间,从而提高匹配场处理的速度和效率,逐步满足定位的作用距离和深度等数据的实时性及精度要求。

参考文献:

[1] GRORGE A, GARCIA J, KIM K. Distributed parallel processing techniques for adaptive sonar beamforming[J]. Journal of Computational Acoustics,2002,10(1):1-23.

[2] REDONDO J L,FERNANDEZ J, GARCIA I, *et al*. Parallel algorithms for continuous competitive location problems[J]. Optimization Methods and Software,2008,23(5): 779-791.

[3] 李建龙,潘翔.不确定海洋环境下的贝叶斯匹配场处理[J]声学学报,2008,33(3):205-211.

[4] 杨坤德. 水声阵列信号的匹配场处理[M]. 西安:西北工业大学出版社,2008:184-186.

[5] PORTER M B. The KRAKEN normal mode program[M]. Italy: SACLANT Undersea Research Centre,2001:194.

[6] 笪良龙,范培勤,陈冬滨. 水平不变海洋声道中 WKBZ 简正波方法的并行算法[J]. 系统仿真学报, 2006,18(7):1980-1983.

[7] 杨坤德,马远良,张忠兵,等. 不确定环境下的稳健自适应匹配场处理研究[J]. 声学学报,2006,31(3):255-262.

[8] TOLSTOY A,HOROSHENKOV K V,BIN ALI M T . Detecting pipe changes via acoustic matched field processing[J]. Applied Acoustics,2009,70(5):695-702.

[9] QUINN M J. MPI 与 OpenMP 并行程序设计[M]. 陈文光,武永卫,译. 北京:清华大学出版社,2004.

[10] BARRY W, MICHAEL A. 并行程序设计[M]. 陆鑫达,译. 北京:机械工业出版社,2005.

(上接第 513 页)行,引入人工智能等方法计算元数据的活跃度,以提高元数据分级的准确性和有效性。

参考文献:

[1] ROSELLI D, LORCH J R, ANDERSON T E. A comparison of file system workloads[C]//Proc of the Annual Conference on USENIX Annual Technical Conference. Berkeley, CA: USENIX,2000:41-54.

[2] PAWLOWSKI B, JUSZCZAK C, STAUBACH P, *et al*. NFS Version 3: design and implementation[C]//Proc of the Summer USENIX Conference.[S.l.]: USENIX,1994:137-151.

[3] MORRIS J H, SATYANARAYANAN M, CONNER M H, *et al*. Andrew: a distributed personal computing environment[J]. ACM Communications of the vol. 29, no. 3, pp. 1986,29(3):184-201.

[4] SATYANARYANAN M, KISTLER J J, KUMAR P, *et al*. Coda: a highly available file system for a distributed workstation environments [J]. IEEE Trans on Computers,1990,39(4):447-459.

[5] CATE V, GROSS T. Combining the concepts of compression and caching for a two-level file system[C]//Proc of the 4th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM Press,1991:200-211.

[6] FLOYD R. Short-term file reference patterns in a Unix environment, Technical Report TR-177[R]. New York: Computer Science Dept., University of Rochester,1986.

[7] RIEDEL E, KALLAHALLA M, SWAMINATHAN R. A framework for evaluating storage system security[C]//Proc of the 1st USENIX Conference on File and Storage Technology. Berkeley, CA: USENIX,2002:15-30.

[8] STAELIN C H. High performance file system design[D]. Princeton: Dept. Computer Science, Princeton Univuniversity,1991.

[9] WEIL S A, POLLACK K T, BRANDT S A, *et al*. Dynamic metadata management for petabyte-scale file systems[C]//Proc of ACM/IEEE Conference on Supercomputing. Washington DC: IEEE Computer Society,2004:4.

[10] CORBETT P F, FEITELSO D G. The Vesta parallel file system[J]. ACM Trans on Computer System,1996,14(3):225-264.

[11] BRANDT S A, LAN Xue, MILLER E L, *et al*. Efficient metadata management in large distributed file systems[C]//Proc of the 20th IEEE / 11th NASA Goddard Conference on Mass Storage System and Technologies. Washington DC: IEEE Computer Society,2003:290-297.

[12] 苏勇,周敬利,余胜生,等. 基于共享存储池的元数据服务器机群的设计研究[J]. 小型微型计算机系统,2007,28(4):734-737.

[13] 许春聪,黄小猛,徐鹏志,等. CarrierFS:基于虚拟内存的分布式文件系统[J]. 华中科技大学学报. 2010,38(1):37-42.

[14] 徐正全,高路,李忠民,等. 基于非结构化 P2P 网络的地形数据存储系统[J]. 武汉大学学报,2010,35(1):122-125.

[15] 董守斌,赵铁柱. 面向搜索引擎的分布式文件系统性能分析[J]. 华南理工大学学报,2011,39(4):7-14.

[16] 吴婷,鞠时光,蔡涛. 基于 DBMS 的元数据管理策略[J]. 计算机应用研究,2010,27(4):1297-1300.

[17] 王雷春,陈世鸿,胡瑞敏. 基于移动 agent 和元数据的位置服务信息查询[J]. 计算机应用研究,2008,25(12):3794-3796.

[18] 刘仲,周兴铭. 基于目录路径的元数据管理方法[J]. 软件学报,2007,18(2):236-245.