刘 军，黄文辉. 交互分析软件MSDP解码效率的优化[J]. 华南地震,2017,37(3):41-47. [LIU Jun，HUANG Wenhui. Decoding Efficiency Optimization for MSDP[J]. South China journal of seismology,2017,37(3):41-47.]

交互分析软件MSDP解码效率的优化

刘 军，黄文辉
(广东省地震局，广州 510070)

摘要：目前的交互分析软件MSDP使用单线程解码，即在解码过程中只有一个CPU处于工作状态，其他CPU始终处于空闲状态。通过对MSDP的改进，利用多线程机制，充分调用多CPU的计算能力，实现了多线程同步解码，从而使得MSDP的解码效率大幅提升，以四核CPU为例，解码时间缩短了近70%。

关键词：交互分析；MSDP；多线程；解码；seed格式
中图分类号：P315.9 **文献标志码**：A **文章编号**：1001-8662（2017）03-0041-07
DOI：10.13512/j.hndz.2017.03.006

Decoding Efficiency Optimization for MSDP

LIU Jun，HUANG Wenhui
(*Earthquake Administrator of Guangdong Province*，*Guangzhou* 510070，*China*)

Abstract：At present, interaction analysis software MSDP uses single thread to decode waveform data, that means, during the decoding process only one CPU is working, the other CPUs remain idle. Through the MSDP improvement, the paper uses the multi-threading mechanism and fully multi-CPU computing power to achieve a multi-thread synchronization decoding, which is allowing the MSDP decoding efficiency increased dramatically, take quad-core CPU as example, the decoding time costing is reduced by nearly 70%.

Keyword：Manual Analysis；MSDP；Multi-thread；Decode；Seed format

0 引言

交互分析软件MSDP是JOPENS系统的一个重要模块，其主要功能是地震震相分析，定位。在地震监测的日常编目工作和速报时都会用到。日常工作中会反复从数据库读取波形，显示波形。在地震速报工作中，是从流服务缓存中读取数据的。速报时需要争分夺秒，MSDP①软件的性能直接关系到速报的效率。这两种情况下读取到的数据最终都要经过MSDP解码处理后，波形才会显示出来，然后才能进行后面的人工交互操作[1]。

随着各个台网的数据共享越来越频繁，几乎每个台网都会共享周边台网的数据，MSDP打开的台站波形数目也越来越多，解码时间也呈线性增

收稿日期：2016-09-06
基金项目：地震行业科研专项(201308008)资助
作者简介：刘 军(1982-)，男，工程师，主要从事地震监测研究。
E-mail：duanmu.lj@qq.com.

① JOPENS用户手册0.5.2. 广东省地震局，2013

长。以广东台网为例，九五期间只需处理 14 个台站左右的数据，十一五增加到 40 多个台站的数据，到现在加上共享的台站总共有 95 个台站的数据量，而且随着监测要求的提高，台站会越来越密，日常及速报需要处理的台站也势必越来越多。对于国家速报备份系统，接入了全球 1 000 多个台站的数据，用 MSDP 以现有的机制打开 1 000 多个台站的数据，哪怕是短短 1 个小时的数据，其消耗的时间也相当可观。

现在计算机的 CPU 趋向于多核化，并发计算处理能力越来越强。我们目前使用的 MSDP 都是单线程解码，即在解码过程中只有一个 CPU 处于工作状态，其他 CPU 始终处于空闲状态。以在 Intel(R)Xeon(R)CPU E5506 @ 2.13GHz 四核处理器工作站运行为例，使用操作系统 FreeBSD 9.0，JAVA 虚拟机版本为 1.6.0，由图 1 可以看出单线程解码时各个 CPU 的工作状态。其中 CPU1，CPU2，CPU3 并没有参与解码，CPU4 的使用率始终在 90%以上。高性能的并发计算能力并没有被充分利用，原本可以节省的时间，可以提高的效率并没有实现。基于此，我们试图充分利用多 CPU 的并发计算能力，对多个台站数据同时解码，使解码时间缩短，提高 MSDP 数据处理能力和效率。

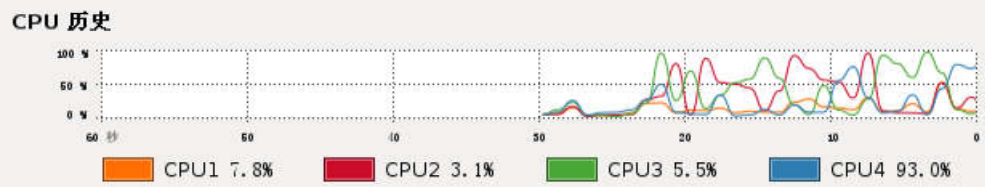

图 1 单线程解码时 CPU 工作状态

Fig 1 Single thread CPU decoding work status

1 现行的解码策略

MSDP 数据处理流程如图 2 所示，从JOPENS 数据库或流服务缓存中读取数据 -> 网络传输 -> MSDP 解码 -> 波形显示。本文的目标是在解码环节进行优化，提升效率。MSDP 处理数据的最基本单元是 mini seed[②]，它是 seed 一种简化格式。每个台站的数据都放在一个链表数组中，每个数据包以 mini seed 格式存储。现有的解码流程是(如图 3 所示)：

(1) MSDP 把从数据库或缓存的数据放在数组矩阵中，每一列代表某一台站的一个通道。

(2) 解码时，先处理第一个通道。

(3) 处理完成后，再处理下一个通道；以此循环，直到所有通道处理完成。

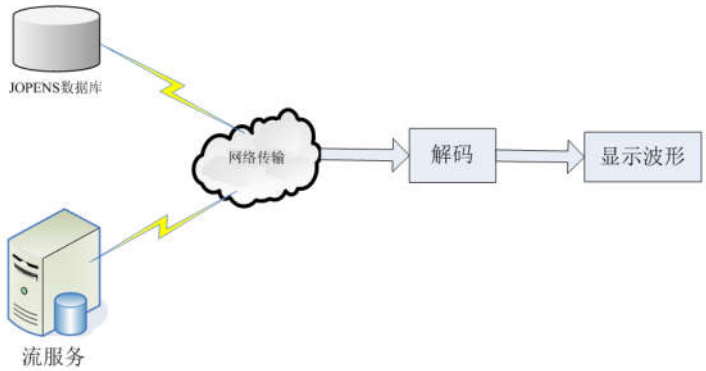

图 2 MSDP 数据流

Fig.2 MSDP data flow

因为解码的工作全都是在一个线程中进行的，解码是个消耗时间的工作，从图 3 所示示意图中我们可以看出，其他 CPU 在第一个 CPU 工作时，却无事可做，因为程序设定为只让一个 CPU 来干解码这种耗时耗力的工作。

2 解码策略分析

为了让所有的 CPU 在解码时都能发挥自己的力

②Timothy K，Ahern. SEED Reference Manual Version 2.4. IRIS，2012

量，我们需要换一种工作方式，就是把解码这个任务分给每个 CPU，让他们同时工作，我们的原则是：在解码期间，尽量让每一个 CPU 不闲着，都有事可做。接下来我们需要考虑如何分工？这是一个很重要的问题。如果分工策略不合适必定会使得某些 CPU 偷懒。因为 seed 格式为动态压缩，每个台站乃至每个通道的压缩率都不是一样的，即使 CPU 同时工作，他们的工作量还是会有差别的。实现并发解码过程中有多重策略，我们需要一一考量，数据处理基本单位有三种情况考虑：一是以数据包作为处理单元，二是以台站作为处理单元，三是以通道作为处理单元。分配解码也有两种方式：第一种是将所有数据事前平均分配，交给每个 CPU 处理；第二种是动态分配任务，每个 CPU 先领一份任务，做完后立即领下一份；下面我们逐一分析这些策略的优劣[3-6]。

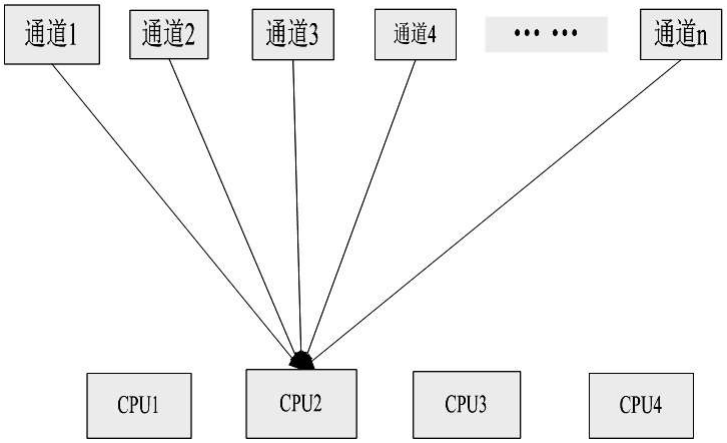

图 3　MSDP 单线程工作模式

Fig 3　MSDP single thread modee

2.1　以数据包作为处理单元

目前 MSDP 定义好的数据组织结构为台站–通道–数据包。如果以数据包作为处理单元，就要重构 MSDP 的数据组织结构，程序改动的工作量相当大，另外每个数据包都是包含数秒至数十秒的波形数据，如果以数据包作为处理单元，解码后数据需要按台站分组，分组好的数据还需要按时间重新排序连接好。这已超出了本文的研究范围，所以以数据包作为处理单元不是一个可行的，有效的方式。我们不打算采用这种细粒度的处理。如图 4 所示为以数据包作为处理单元来解码的工作模式。

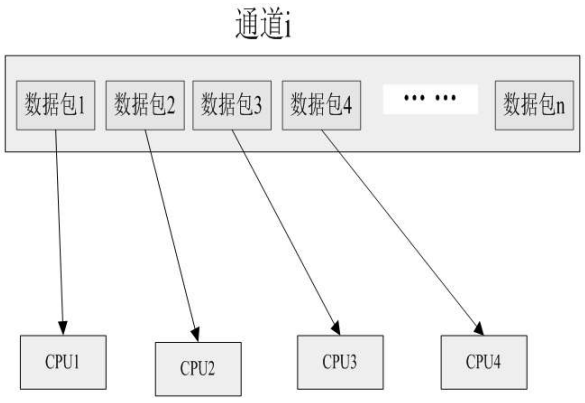

图 4　以数据包作为处理单元

Fig.4　Using data packets as processing unit

2.2　以台站作为处理单元

每个台站一般有三个通道，也有些台站有六通道乃至更多。若台网的数据都是三通道的还好些，分配给每个 CPU 的工作量大致相同，每个

CPU 都能一直处于工作中。但是如果台网的台站既有三通道的，又有六通道的，此时再以台站为单位分配，就不合适了。解码六通道必定比解码三通道要消耗更多的时间。这样又会使得分工不均。如图 5 所示为以台站作为处理单元来解码的工作模式。

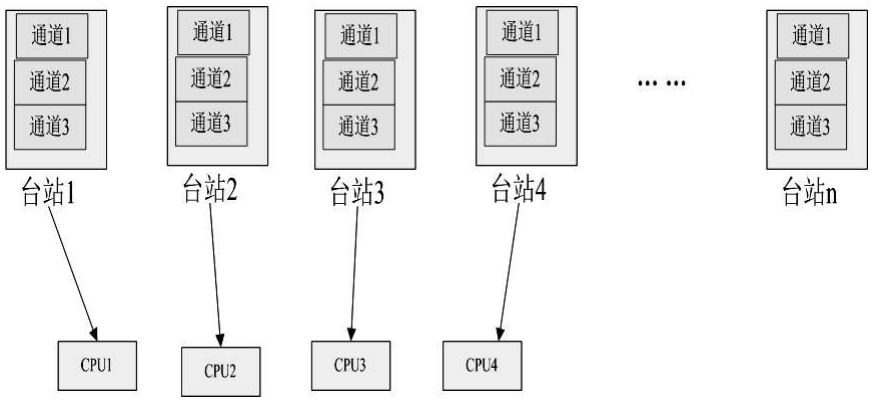

图 5 以台站数据作为处理单元

Fig.5 Using station data as processing unit

2.3 以通道作为处理单元

这种处理方式就可以有效的解决之前两种方式所遇到的问题。以通道作为处理单元解码后的数据是一个完整的，有序的数据，可以直接用来显示波形，程序结构几乎不用修改；另外对于通道数不同的台站处理也没有问题，我们不管台站是几通道的，忽略台站的区别，以通道为单位，先处理完一个个通道，然后再按台站分组显示，这样代码修改的工作量也不会太大。本文就采用以通道作为处理单元的方式来解码。如图 6 所示为以通道作为处理单元来解码的工作模式。

注意我们在这里假定每个台站的数据采样率都是一致的，如果采样率不一样，即使每个通道的数据量也会有很大的不同。处理的方式就会更为复杂

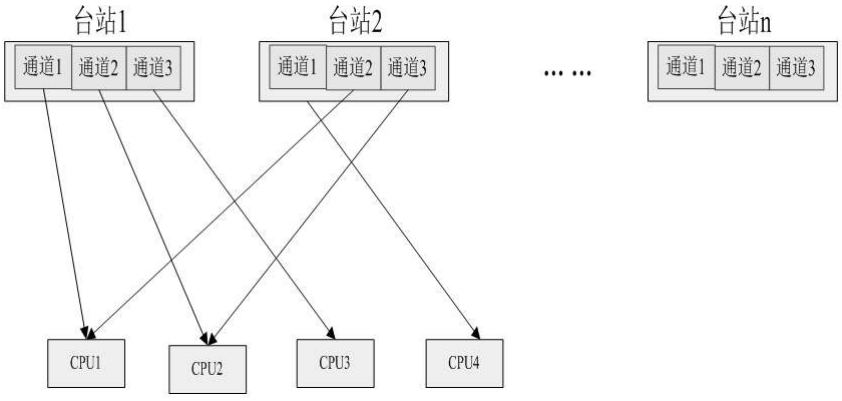

图 6 以通道数据作为处理单元

Fig.6 Using channel data as processing unit

3 解码分配方式

当我们确定了要以通道作为处理单元的方式来解码后，接下来就要确定如何分配解码，有两种方式可供我们选择：

3.1 静态分配

第一种是将所有数据事前平均分配，交给每个 CPU 处理。这种处理方式有一个潜在的问题，这是由于 seed 的动态压缩特性造成的，每个通道都是动态压缩，如果台站噪声过大，数据要缩率就比较低，数据量就会较大。CPU 解码就要花更多的时间；如果事先把任务都分配好，表面上看起来时平均分配，可是其实真实的工作量还是不一样的。有些 CPU 要处理的数据量小，处理完就又会闲下来无事可做。使用这种方式的前提是所

有台站的数据量是一致的，由于 seed 的动态压缩，这个假设是不成立的。如图 7 所示为 CPU 平均分配解码工作量。

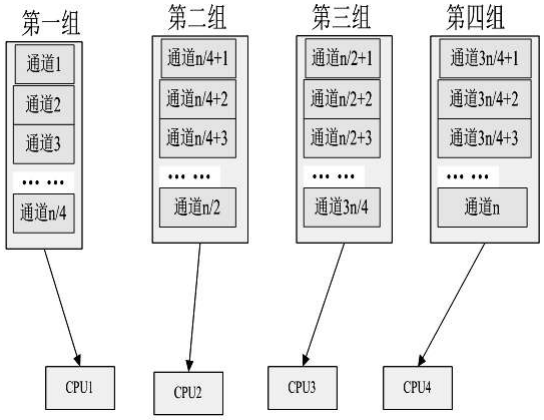

图 7　平均分配解码工作量

Fig.7　The workload of decode equally distributed

3.2　动态分配

第二种是动态分配任务，每个 CPU 先领一份任务，做完后立即领下一份。这种方式的特点是每次给每个 CPU 分配一定量的解码工作，不多也不少，干完了还有活可干，不会闲下来，这比第一种方式就好多了。在解码工作期间，保证了所有的 CPU 都处于工作状态。我们就以这种最优的方式来解码。如图 8 所示为 CPU 动态分配解码工作量

3.3　使用线程池解码

多次创建和销毁线程会消耗加大的系统资源，使用线程池解码可以提高硬件资源的使用效率，相应的代码如下：

```
ChanLoader cloader = new ChanLoader (uniSTms, uniEDms, fileName);
cloader.init (numThreads); //初始化线程池
ChannelWaveform cwf = seedVolumePlugin.getNextChannelWaveform ();
if (cwf == null)
return;
do {
loader.loadSampData (cwf) ; //使用多线程解码，每次处理一个通道
} while ( ( cwf = seedVolumePlugin.getNextChannelWaveform ())! = null) ;
cloader.destroy (); //销毁线程池
```

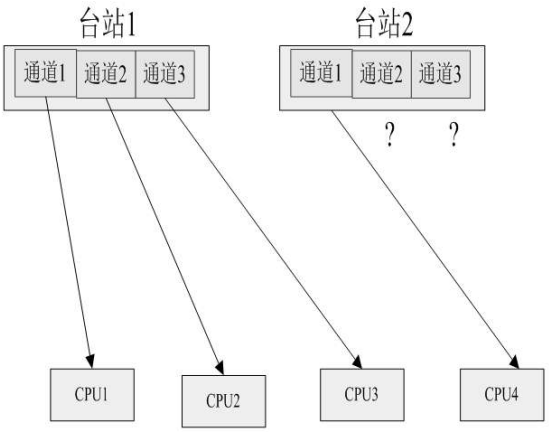

图 8　动态分配工作量

Fig.8　Dynamic distribution of workload

4　讨论

我们通过对比发现，多线程能够使得多个 CPU 同时工作，不至于只有一个 CPU 工作，其他 CPU 无事可做，如图 9 所示可以看出，多线程机制下，CPU1、CPU2、CPU3、CPU4 的使用率分别为 76.9%、86.2%、86.9、98.5%，其使用率远远高于单线程时各个 CPU 的使用率。表格可以可以对比出单线程和多线程的工作效率之区别。

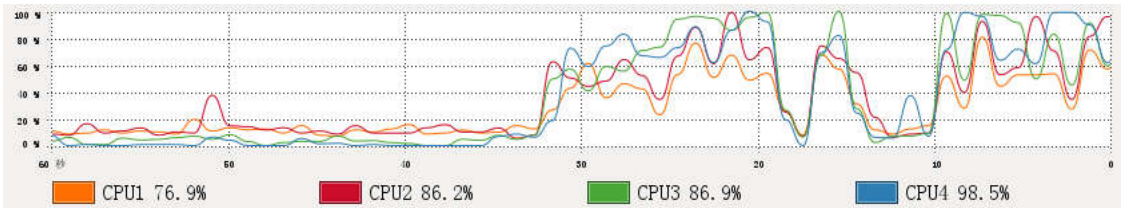

图 9　多线程时各个 CPU 的工作状态

Fig 9　Multithreading work status of each CPU

表 1 为单线程解码的测试结果，测试方法为选取广东台网及周边台站共计 95 个台站的数据进行解码，分别测试 1 h，2 h，3 h，4 h，5 h 的解码耗费时间。每一个时间段都测试 5 次，最后求平均值作为最终测试结果。

表 1 单线程解码

Table 1 Using single thread to decode

时间长度/h	第一次解码耗时/ms	第二次解码耗时/ms	第三次解码耗时/ms	第四次解码耗时/ms	第五次解码耗时/ms	平均耗时/ms	数据包总数/个
1	13 815	18 761	23 545	28 776	19 032	20 785.8	362 554
2	44 156	41 065	28 894	50 229	36 239	40 116.6	718 012
3	66 297	61 288	57 018	44 014	45 587	54 840.8	1 071 813
4	70 399	65 510	66 219	74 755	86 064	72 589.4	1 424 984
5	97 720	79 730	91 381	76 809	91 114	87 350.8	1 776 602

表 2 为多线程解码[9]的测试结果，测试方法为选取广东台网及周边台站共计 95 个台站的数据进行解码，分别测试 1 h，2 h，3 h，4 h，5 h 的解码耗费时间。每一个时间段都测试 5 次，最后求平均值作为最终测试结果。

表 2 多线程解码

Table 2 Using multiple threads to decode

时间长度/h	第一次解码耗时/ms	第二次解码耗时/ms	第三次解码耗时/ms	第四次解码耗时/ms	第五次解码耗时/ms	平均耗时/ms	数据包总数/个
1	5 024	3 887	4 006	4 803	4 125	4 369	362 554
2	7 125	6 279	6 965	7 054	7 570	6 998.6	718 012
3	10 517	11 161	15 948	20 217	16 281	14 824.8	1 071 813
4	20 235	15 917	16 972	19 248	16 615	17 797.4	1 424 984
5	36 256	37 954	23 420	24 810	24 258	29 339.6	1 776 602

表 3 为单线程与多线程解码效率对比数据，由表中节省时间一栏可以看出每个时长的解码效率都是多线程解码远远高于单线程解码。图 10 为单线程与多线程解码效率对比柱图。

表 3 单线程与多线程解码效率对比数据

Table 3 Comparison data of single-threaded and multi-threaded decoding efficiency

时间长度/年	单线程解码/ms	多线程解码/ms	节省时间/%
1	20 785.8	4 369	79.0
2	40 116.6	6 998.6	82.6
3	54 840.8	14 824.8	73.0
4	72 589.4	17 797.4	75.5
5	87 350.8	29 339.6	66.4

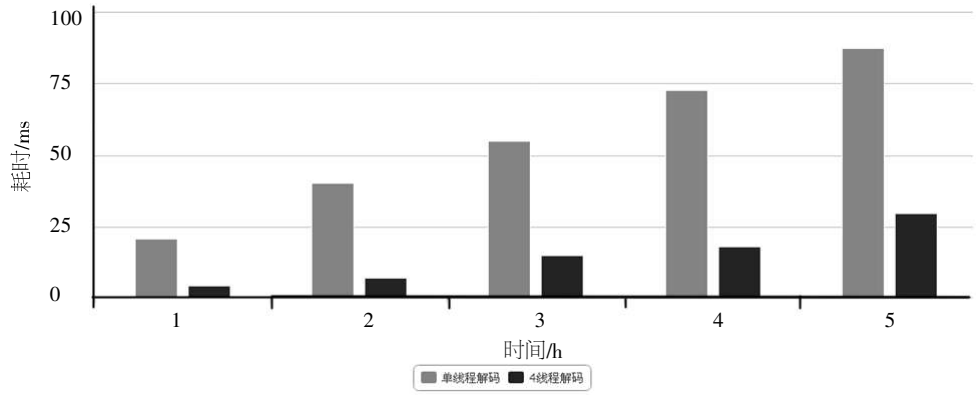

图 10 单线程与多线程解码效率对比柱状图

Fig.10 Comparison histogram of single-threaded and multi-threaded decoding efficiency

5　结语

通过使用多线程机制，不仅可以充分利用现有的硬件条件，而且可以极大的提高 MSDP 解码的效率，这对 MSDP 整体性能的提升无疑有积极的影响。在 jopens-0.5.2 版里，已经正式将多线程解码功能添加到了新版本的 MSDP 里。相信有了多线程解码，无论是地震速报，还是日常地震波形的浏览分析，都会节省不少时间。

参考文献：

[1] 刘　军，康　英. 新版测震台网地震观测报告[J]. 华南地震,2015,35(4):25-30

[2] 结城浩. JAVA 多线程设计模式[M]. 北京:中国铁道出版社，2005

[3] 王　挺,陈修吾,叶佳宁. 基于自动地震速报的地震应急基础信息快速提取模块的研究与实现[J]. 华南地震，2016,36(1):16-23.

[4] 刘　军，黄文辉. 测震台网业务交换平台[J]. 华南地震，2016,36(1):36-43.

[5] 刘　军,康　英. JOPENS 编目检查系统设计[J]. 华南地震，2016,36(2)：89-95.

[6] 黄文辉,沈玉松,吕作勇,等. 地震超快速报系统试运行结果评估[J]. 华南地震,2016,36(4):1-7.