基因序列分析软件 Hmmpfam 的可扩展并行性能优化*

陈 军[1,2+], 赵文辉[2], 莫则尧[1], 李晓梅[3]

[1](应用物理与计算数学研究所 高性能计算中心,北京 100088)

[2](国防科学技术大学 计算机学院,湖南 长沙 410073)

[3](总装备部指挥技术学院,北京 101416)

Scalable Parallel Performance Optimization of the Gene Sequence Analyzing Software Hmmpfam

CHEN Jun[1,2+], ZHAO Wen-Hui[2], MO Ze-Yao[1], LI Xiao-Mei[3]

[1](High Performance Computing Center, Institute of Applied Physics and Computational Mathematics, Beijing 100088, China)

[2](School of Computer, National University of Defense Technology, Changsha 410073, China)

[3](Institute of Command and Technology of Equipments, Beijing 101416, China)

+ Corresponding author: Phn: +86-10-62014411 ext 2670, E-mail: chenojun@yahoo.com

Received 2003-03-24; Accepted 2003-05-27

Chen J, Zhao WH, Mo ZY, Li XM. Scalable parallel performance optimization of the gene sequence analyzing software Hmmpfam. *Journal of Software*, 2004,15(2):170~178.

http://www.jos.org.cn/1000-9825/15/170.htm

Abstract: A scalable parallel MPI (message passing interface) version of the popular protein structure prediction tool Hmmpfam is presented, which is one of the kernel programs in the HMMER package. The master process in the previous PVM (parallel virtual machine) version is a communication bottleneck, and the speedup will decrease rapidly when running on large scale parallel systems. A novel three-level communication structure is presented, by which the parallel processing at sequence level and HMM model level is obtained in both. Meanwhile, the load-balance strategies to sequence level and HMM model level distribution are provided separately. Since disk access for getting HMM model costs very much, a so-called "once load" strategy is provided to reduce the cost. By all these optimization methods, 95% in parallel efficiency is achieved when running on a parallel computer containing more than 700 processors.

Key words: parallel computing; gene sequence analysis; HMMER

摘 要: 基于 MPI(message passing interface)平台实现了 HMMER 软件包核心程序之一 Hmmpfam 的大规模并行

* Supported by the National Grand Fundamental Research 973 Program of China under Grant No.G199508032805 (国家重点基础研究发展规划(973)); the National High-Tech Research and Development Plan of China under Grant No.863-2002AA104570 (国家高技术研究发展计划(863))

作者简介: **陈军**(1974－),女,湖南衡阳人,博士,副研究员,主要研究领域为并行算法,性能评价;**赵文辉**(1972－),男,博士,助研,主要研究领域为软硬件协同设计;**莫则尧**(1971－),男,博士,研究员,主要研究领域为并行算法;**李晓梅**(1938－),女,教授,博士生导师,主要研究领域为并行算法,可视化.

计算.该版本针对原 PVM(parallel virtual machine)并行版本在并行规模扩大后,master 易成为通信瓶颈的问题,对通信结构进行了优化,提出了一种新的三层通信结构,在序列和 HMM 模型的两个层次上实现了并行化,并分别提供了有效的负载平衡策略,同时优化了 I/O 性能,在 700 多台处理机上达到 95%的效率.

关键词: 并行计算;基因序列分析;HMMER

中图法分类号: TP311 **文献标识码**: A

利用计算机进行序列分析是当今生物信息学领域中常用的研究手段,通过将新测定的序列和数据库中已知功能的序列进行相似性比对,找出具有相同残基的功能位点,确定新测定序列与数据库中已知结构和功能的序列间的相似性关系,从而在一定可信度的情况下确定新序列的结构和功能信息[1].HMMER[2]是一种采用隐式 Markov 模型(hidden markov models,简称 HMM)[3,4]来进行敏感性数据库搜索的序列分析工具,由华盛顿大学医学院的 Sean Eddy 开发,共有 16 个独立程序,其中 Hmmpfam 是使用最为广泛的程序之一.HMM 模型描述了大量相互联系的状态之间发生转换的概率,本质上是一条表示匹配、缺失或插入状态的链.Hmmpfam 程序利用一个或多个序列搜索剖面隐式 Markov 模型(profile HMM)组成的库(例如广泛使用的 Pfam),检测序列比对结果中的保守区,从而识别出这些序列中已知的蛋白质或核苷酸结构域,进而可以详细阐明序列间的关系,包括超家族、家族、亚家族和种属特异等不同水平的关系.

计算时间长是序列分析面临的主要问题,如何利用并行计算手段缩短序列分析的时间成为当前的研究热点.SGI 公司针对生物计算提出了高吞吐率计算环境(high throughput computing environment,简称 HTC),并将 HMMER 作为典型应用,用来衡量 SGI Origin 300 的性能[5].苹果公司采用 AltiVec 技术,针对他们的处理器实现了 HMMER 的向量化加速版本[6].这些公司主要针对他们的硬件结构而不是从算法角度来优化程序,目前只给出了小规模系统上的性能.全球最大的网格软件开发商 Platform 计算公司利用网格计算环境运行 HMMER 程序,宣称他们使用 5 000 个 Platform ActiveCluster 节点,用序列数据库 SWISS-PROT(大约 80 000 条序列)在 PFAM(包含 2866 个 HMM 模型)上搜索约需 2 个小时[7].但是,作为商业秘密,这些公司没有公布程序优化的细节.

本文针对 hmmer-2.2g(HMMER 的当前最新版)中的 Hmmpfam 程序进行了性能优化,并实现了 MPI(message passing interface)版本;在一台并行机(以下称 PPC)上采用缺省参数值运行,取得了较好的加速效果.本文第 1 节简单介绍 Hmmpfam 程序;第 2 节给出具体的性能优化方法;第 3 节是实验结果分析;最后总结全文.

1 Hmmpfam 描述

Hmmpfam 已提供了 Pthreads 和 PVM(parallel virtual machine)[8]并行版本.但是,基于 Pthreads 开发的程序是不可扩展的,而 PVM 程序采用如图 1 所示的树状通信结构:1 个主进程,n 个子进程.主进程负责从序列文件读出一条序列,将它广播给所有子进程;然后,依次将数据库中所有参加比对的 HMM 模型的序号发送给空闲的子进程.各子进程接到任务后,利用序号在数据库中定位到 HMM 模型的位置并进行比对,然后将结果返回给主进程.当数据库中所有 HMM 模型和该序列的比对完成后,主进程根据得到的局部比对计分值和全局比对计分值产生局部比对队列 D 和全局比对队列 G,并按计分值由高到低分别对其进行排序.然后,主进程继续读入下一条序列,重复上述步骤,直到处理完所有序列为止.Master 和 slave 之间以任务池方式来交换数据:任务池位于 master 节点上,Slave 从任务池中取出 HMM 模型序号,并放入比对结果.对于每一条序列,都要多次进行放置和获取操作.

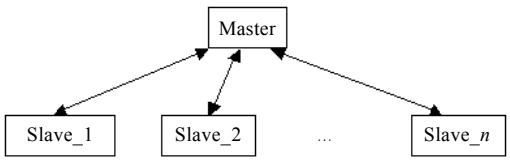


Fig.1 Level-2 tree communication structure between master and slave process

图 1 主进程与子进程间的两层树状通信关系

由于 MPI[9]已成为消息传递工业标准,有必要将 PVM 程序移植到 MPI 上来,以便在大多数并行机上运行.初始阶段,我们沿用原有并行策略进行移植.在实际测量中,我们发现,当处理机增加到一定规模时,性能会显著

下降.在对 6 000 条序列进行搜索时,用 100 台处理机的时间与用 512 台处理机的时间基本相同.通过对原程序进行分析发现,这是由于原并行结构可扩展性差引起的.在上述通信结构下,特别是在应用 Bcast 通信原语时,当处理机规模增加到一定数目时,master 易成为通信瓶颈.因此,有必要重新组织合理的通信结构.图 2 显示了 PPC 平台上 MPI_Bcast 的性能.从图中可以发现,当 $size \le 10^5$ 且 $p>64$ 时,随着 p 的增加,广播时间开销增长较快.在 Hmmpfam 中,广播的字节大小主要由序列长度决定,而目前测试的序列大部分长度不大于 10^5,因此有必要重新组织合理的通信结构.

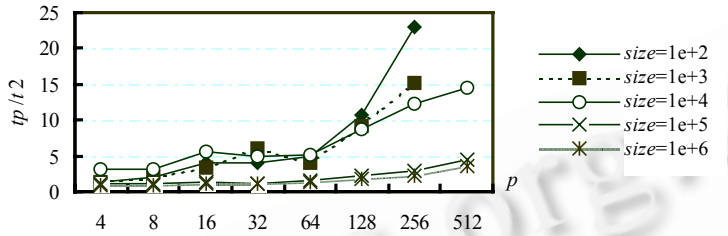


Fig.2 MPI_Bcast performance on PPC, where *size* is the message length, *tp* and *t*2 are the broadcast time on *p* and 2 processors respectively

图 2 PPC 上 MPI_Bcast 的性能(*size* 为消息长度,*tp* 和 *t*2 分别为在 *p* 台处理器和 2 台处理器上的广播时间,纵轴为 *tp* 与 *t*2 的比值)

另外,假设有 *num* 个 HMM 模型要搜索比对,对于每一条序列 *seq*,master 首先广播该序列的信息以及计分阈值 *thresh*(计分值高于这些阈值的比对结果才被输出),然后将发送 *num* 条仅包含模型序号(1 个整型数)的消息,并接收 *num* 条包含计分值 *sc*,*evalue* 和路径 *tr* 的消息.假设两个节点间发送/接收 *m* 个单位数据(如字节)的通信开销用下式表示:

$$t = O + g \times m \tag{1}$$

其中 O 为通信启动开销,g 为发送一个单位数据的间隔时间(通信带宽的倒数),则 master 对序列 *seq* 的通信开销见表 1,其中 l_f, l_d, l_i 分别表示一个单精度、双精度浮点、整型数所含字节数,t_l 是序列长度和 HMM 模型长度的函数.

Table 1 Prediction of communication time the master needed to analyze one sequence *seq* in the previous program

表 1 原程序中 master 分析 1 条序列 *seq* 所需通信时间估计

Communication primitive	Number	Time prediction
Bcast(*seq*,*thresh*)	1	–
Send(*nHMM*)	*num*	$num\times(O+g)$
Recv(*sc*,*evlaue*,*tr*)	*num*	$num\times[O+g\times(l_f+l_d+t_l\times(2l_i+1))]$

在现代机群和 MPP 系统中,软件开销占通信时间的主导地位[3].原程序频繁地在 master 和 slave 之间进行小数量的通信,使得通信端产生的开销很大.由于某些并行机对长消息提供了专门的硬件支持,因此应减少通信的频率,加大一次通信的通信量.

原程序只开发了 HMM 模型级别上的并行,即只实现了一条序列在多个 HMM 模型的并行搜索.我们在消除了原程序中序列之间的伪相关以后,实现了序列与序列之间的并行处理.另外,当处理机数目增加到一定规模之后,可以考虑将 HMM 数据驻留在内存中,以减少访问硬盘的次数.在原程序中,为了实现 HMM 模型搜索之间的负载平衡,采用任务池方式,动态地决定每个进程所分配的 HMM 模型.这样,各子进程无法预知将处理的 HMM 模型的情况,只能在整个数据库中进行搜索.对于数据库比较大的情况,每个进程的内存不足以容纳整个库的内容,只有访问硬盘,读取所需的 HMM 信息.

综上所述,本文从 3 个方面进行了优化:重组通信结构、引入二级负载平衡策略以及 I/O 性能优化.

2 Hmmpfam 的并行优化

2.1 三层树状通信结构

首先,为了缓解 master 造成的性能瓶颈,我们引入了中间处理层,形成 3 层树状结构.多个节点组成一组,完成对一个序列的分析;每组中有一个管理员节点,负责收集组内其他节点的结果,并根据结果,将有意义的局部比对计分值和全局比对计分值(即计分值大于阈值)分别加入到列表 D 和 G 中.然后,分别对两个列表排序,产生该序列的比对结果,并发送给总控进程.总控进程接收到以后,作输出处理,若有未作分析的序列,将该序列发送给该组.重复此过程,直到处理完所有序列为止.进程之间的通信结构如图 3 所示.

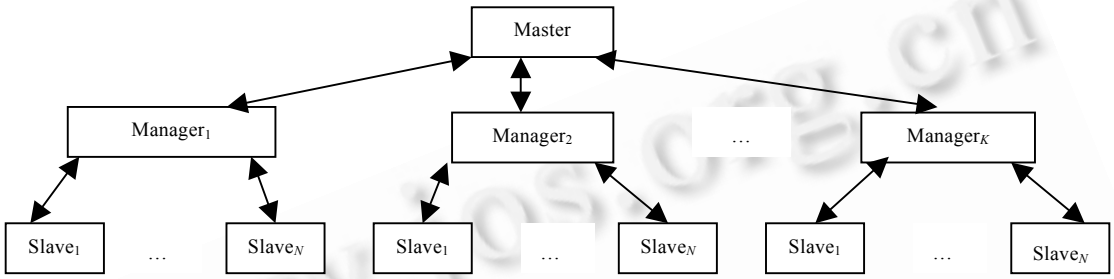


Fig.3 Modified communication structure between processes

图 3 改进的进程间通信结构

假设作业分配方式如下:1 个总控进程 master,其余节点分成 K 个组,每组包含 N 个 slave 和 1 个 manager,则共需 $1+K\times(N+1)$个处理机.序列文件放置在 master 节点上,而 HMM 数据库存储于 manager 和 slave 节点的局部硬盘中.各组的 manager 以任务池方式从 master 获得任务,即当某个 manager 空闲时,将通知 master 以获取新的序列.在上述通信结构下,系统可以实现两级并行:(1) 序列间并行,即不同组处理的序列不同;(2) 对于某特定序列,组内各 slave 之间的并行.显然,当 $K>1$ 时,该通信结构可以有效地加快多个序列的处理速度.当 $K=1$ 时,三层通信结构退化为只实现 HMM 模型级别的并行,序列间是串行处理的.

其次,为了减少通信频率,增大一次通信的数据量,我们在组内采用如下通信方式:manager 预先为组内各 slave 分配搜索范围,用 HMM 模型在数据库中的序号范围[from,to]表示;对于 master 发来的任务,manager 将 *seq* 广播给组内 slave,等待每个 slave 发送来的结果(可能是一次或多次);slave 收到 *seq*,根据自己负责的范围,到数据库中搜索序号在[from,to]之间的 HMM 模型进行比对,将结果打包到 buffer 中,发给 manager;若 buffer 超过用户规定的长度,就分成多次发送.

综上,本文在通信结构方面对原程序进行了以下优化处理:(1) 引入了序列间并行;(2) Slave 预先确定了要搜索的 HMM 模型在数据库之间的位置范围,这为下面的 I/O 优化打下了基础;(3) 原版本中无论比对计分值是否高于阈值,slave 均需将比对结果发送给 master.新版本中仅将高于阈值的比对结果发送给 manager(对于所有 HMM 模型的计分值均小于阈值的情况,发送给 manager 一个标志信号),从而减少了通信量.图 4 给出了数据的流向.实验显示,该方法减少了数据发送的次数,同时,在 master,manager 和 slave 之间实现了部分的流水处理.

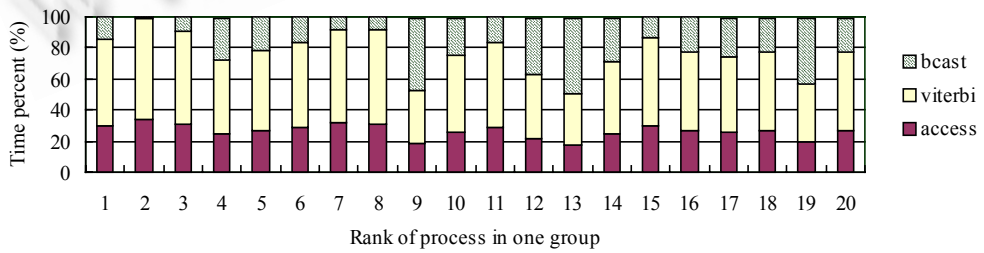


Fig.4 The percent of primary time-consume components in each slave of a group (200 sequences, and p=64, K=3, N=20)

图 4 各组中每个 slave 上主要费时部分占自身总时间的比重(200 条序列,p=64,K=3,N=20)

2.2 重组通信结构后的性能分析

应用上述方法,当用 100 台处理机处理同样的 6 000 条序列时,处理时间不到原版本时间的 1/3.用 vts3_fasta(蛋白质序列文件)的前 200 条序列对 Pfam_ls 库(包含 3 735 个 HMM 模型)进行搜索,在处理机数目不同情况下的并行效率和吞吐率如图 5 所示.这里,吞吐率为平均每秒处理的序列数目;并行效率 $E=T_1/(T_p\times p)$,其中 T_1 和 T_p 分别是串行时间和 p 台处理机上的并行时间,串行时间在原程序的串行版本上测得.另外,本文中所有并行执行均采用 Hmmpfam 的缺省参数值.当在 100 台处理机上运行时,并行效率为 97%.当处理机数达到 703 时,效率仍可以达到 75.14%,平均每秒处理 2.802 条序列.当对 vts3_fasta 的所有序列(103 807 条序列)在 Pfam_ls 上进行搜索时,在 703 台处理机上的速度达到平均每秒 3.96 条序列.

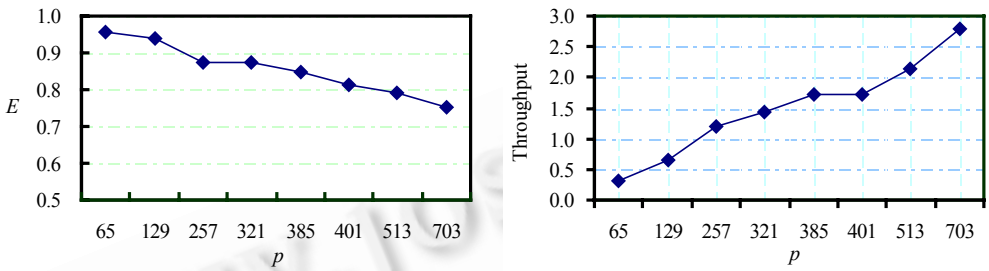


Fig.5 Parallel efficiency and throughput of the modified Hmmpfam on PPC (the number of sequence is 200)

图 5 在 PPC 上优化后的 Hmmpfam 的并行效率和吞吐率(序列数为 200)

当 N 取不同数值时,获得的并行效率不同.图 6 显示了当 N+1 为 8,16,32,64,128 时,该 200 条序列分别在 129,321,513 台处理机上的运行效率.

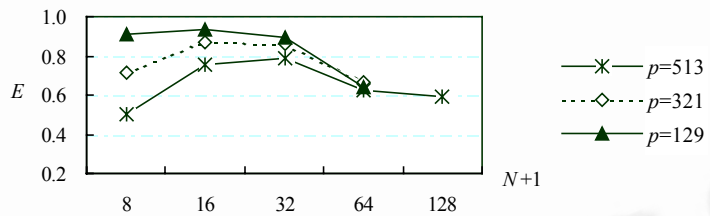


Fig.6 Parallel efficiency with different N

图 6 N 值不同时的并行效率

从图 6 和表 2 可以看出,当 N+1=16 或 32 时,性能较好.另外,在 K 和 N 的取值上应作权衡.这是因为,当处理机数目固定时,若 N 过大,每个 slave 处理的 HMM 模型就少,计算时间也少,而 manager 要接收消息的次数增加了,通信时间随之而增加;同时 K 相应减少了,序列的处理速度降低,从而降低了性能.另外,若 K 过大,则 N 相应变小,每个 slave 处理的 HMM 模型就多,计算时间增加了,同时要传输的信息量也增加了,一旦信息量超过用户设定的缓冲区大小,就需要多次发送,这同样增加了通信时间,从而导致性能降低.因此,K 和 N 的值应比较接近.

Table 2 Parallel efficiency with different N

表 2 取不同 N 值时的并行效率

p	N+1	K	E (%)
129	16	8	93.7
	32	4	89.22
321	16	20	87.22
	32	10	86.14
513	16	32	75.8
	32	16	78.9

2.3 二层负载平衡策略

定义 1. 分别定义组间负载不平衡系数 η_{inter} 和组内负载不平衡系数 η_{intra} 为

$$\eta_{\text{inter}}=\frac{T_{\max}^{\text{inter}}-T_{avg}^{\text{inter}}}{T_{\max}^{\text{inter}}},\quad \eta_{\text{intra}}=\frac{T_{\max}^{\text{intra}}-T_{avg}^{\text{intra}}}{T_{\max}^{\text{intra}}}.$$

其中, $T_{\max}^{\text{inter}}$ 表示不同组中 slave 的 cpu 时间总和的最大值; $T_{\max}^{\text{intra}}$ 表示同组内 slave 的 cpu 时间的最大值;带下标 *avg* 的量表示平均值.负载不平衡系数介于 0,1 之间,越靠近 1,表示负载越不平衡.

表 3 给出了 200 条序列的测试数据.可以发现,组间平衡性很好,这主要归功于采用任务池获取序列的方式;但是,组内平衡性较差,需进一步优化.图 4 显示了其中一组各 slave 节点上 bcast 部分、viterbi 部分和 access 部分占该 slave 整个时间的比重.其中 access 是定位以及从硬盘读入 HMM 模型信息的时间,viterbi 部分是实现 viterbi 算法的时间,bcast 部分指的是广播通信时间以及由于负载不平衡引起的等待时间.这 3 部分占据了 slave 节点上整个时间的 99%.其中 viterbi 是主要的计算部分,对于同一条序列,处理该序列的同一组 slave 的计算时间将近似正比于它所处理的 HMM 模型长度的总和.为此,我们在组内加入重分配机制,在每一组的所有 slave 处理完 *n* 条序列(这里 *n*=1)以后,根据它们在每个 HMM 模型上消耗的计算时间,重新分配它们要比对的 HMM 模型区域,使同组中 slave 的计算时间尽可能地接近,从而达到负载平衡的目的,见表 4.

Table 3 The inter-group and intra-group imbalance factors when processing 200 sequences

表 3 200 条序列时,组间负载不平衡系数和组内负载不平衡系数

p	*K*	*N*	η_{inter}	η_{intra}								
				Group1	Group2	Group3	Group4	Group5	Group6	Group7	Group8	Group9
64	3	20	0.015	0.225	0.229	0.256	–	–	–	–	–	–
64	7	8	0.057	0.140	0.140	0.082	0.194	0.094	0.149	0.088	–	–
100	3	32	0.053	0.242	0.241	0.171	–	–	–	–	–	–
100	9	10	0.046	0.143	0.182	0.201	0.182	0.139	0.143	0.140	0.162	0.146

Table 4 The inter-group and intra-group imbalance factors after redistribution when processing 200 sequences

表 4 200 条序列分析,采用重分配机制后的组间负载不平衡系数和组内负载不平衡系数

p	*K*	*N*	η_{inter}	η_{intra}						
				Group1	Group2	Group3	Group4	Group5	Group6	Group7
64	3	20	0.02	0.075	0.09	0.081	–	–	–	–
64	7	8	0.036	0.05	0.03	0.04	0.04	0.08	0.041	0.036

图 7 给出了重分配后的情况,同组中各个处理机较为平衡,通信等待开销减少,从而 bcast 部分所占比例减少.同样情况下,采用重分配机制后,在 64 台处理机上总的执行时间减少了 8%.

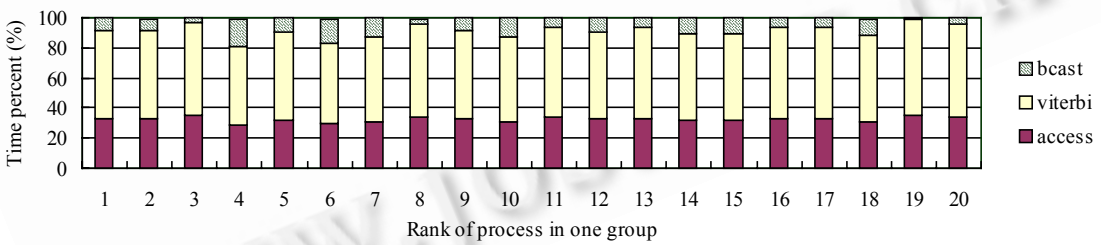


Fig.7 The load balance situation of slaves in the same group after redistribution

图 7 重分配后同组中 slave 的负载平衡情况

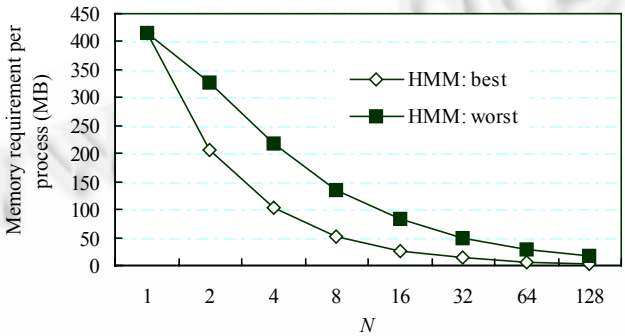
2.4 I/O优化

在上述优化策略下,每个 slave 将在一个相对固定的 HMM 模型序号范围内进行比对计算.此时,可以采用“一次载入”机制实现 I/O 优化,即每个 slave 将涉及的 HMM 模型数据驻留在内存中,与不同序列作比对,减少磁盘访问次数.由于生物计算程序对内存的需求较大,下面将就多个 HMM 模型分析对应的 HMM 数据结构,确定其对内存容量的需求.对常用的 HMM 库 Pfam_ls 的分析结果见表 5.其中,“动态内存容量估计”列出了动态分配内存比例最大的两个部分,viterbi 容量指的是 viterbi 算法中对于每个 HMM 模型需动态申请的工作矩阵空间估计,tr 容量指的是对每个 HMM 模型有意义的比对信息 tr 的大小.这两部分与序列的长度、HMM 模型长度以及预定义最大值(程序预定义为 32MB)相关.“HMM 数据结构所需内存容量估计”列出了所有模型的总和(总值)、平均每个模型所需容量(平均)、不同模型所需容量的最大值和最小值.

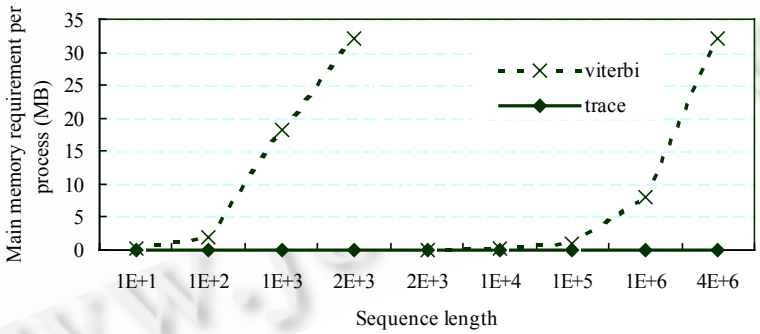
Table 5 Hmmpfam memory requirements prediction
表 5 Hmmpfam 所需内存容量估计

Database	# & maxlenth of HMMs	Dynamic memory requirements prediction per HMM		Memory requirements of data structure of all the HMMs (bytes)			
		viterbi requires	tr	Total (MB)	Average (KB)	Max. (KB)	Min. (KB)
Pfam_ls	3 735 & 1 515	If ≤32MB, then $O(LM)$ else $O(L+M)$	$O(L)$	416	111	726	3

图 8 中的 HMM:best 表示 HMM 平均存储量与进程上分配的 HMM 个数的乘积,而 HMM:worst 表示最差情况下存储量需求最大的前几个 HMM 均分配到某进程上.最坏情况下,当 N=16,32,64,128 时,单进程上 HMM 结构所需空间分别为 81MB,48MB,28MB 和 16MB.图 9 中 viterbi 工作矩阵所需空间在长度为 1 751 处有一个局部最大值,这是由于程序中设置了一个阈值(32MB),当序列长度过长使得 viterbi 算法所需空间超过该值时,则将该序列拆分成多个片段,分别对它们采用 viterbi 算法.若拆分所得片段仍超过阈值,则递归地拆分该片段.片段所需的存储空间可重复利用,但增加了计算时间.



(a) The hmm structure memory requirement prediction on single process when searching into Pfam_ls database using the once load strategy. Generally, it is in the field between the two curves HMM:best and HMM:worst
(a) 若用“一次载入”机制,则采用 Pfam_ls 库时单进程上 hmm 结构存储量估计
(一般落在 HMM:best 和 HMM:worst 两条曲线包围区域中)



(b) The upper memory requirements of viterbi and tr structure on single process when aligning with the HMM model of maximum length (1 515) in Pfam_ls database
(b) 采用 Pfam_ls 时单进程上用于 viterbi 和 tr 结构的存储量上限(这里针对模型长度最大(1 515)的 HMM 模型进行估计

Fig.8
图 8

Pfam_ls 中模型长度最大值为 1 515,最小值为 5.图 9 显示了在 Pfam_ls 上 viterbi 所需内存量的估计,其中 domain 为不同模型长度上达到最大存储量的序列长度区域,即序列长度处于 domain 区域的可以达到 viterbi 存储量需求最大值.表 6 给出了用来比对的序列文件中长度落于 domain 区域中的序列个数占文件中总序列个数的比重.其中 200_seq.fasta 是 vts3.fasta 的前 200 条序列组成的文件,也就是上面我们曾用来测试性能的序列文件.可以看出,对于 vts3.fasta 序列文件来说,viterbi 所需存储量达到上限(32M)的不多.在未使用我们提出的“一次载入”HMM 结构之前,viterbi 存储量占主导地位.另外,序列长度落在 domain 区域内,表明它被拆分的几率较高,

这意味着它的计算时间也增大.表 6 反映了前 200 条序列中长度落在 domain 范围内的序列占总序列个数的比重比相应的 vts3.fasta 要高,这可以部分地解释在 703 台处理机上处理 vts3.fasta 为什么比 200_seq.fasta 的吞吐率要高的问题.

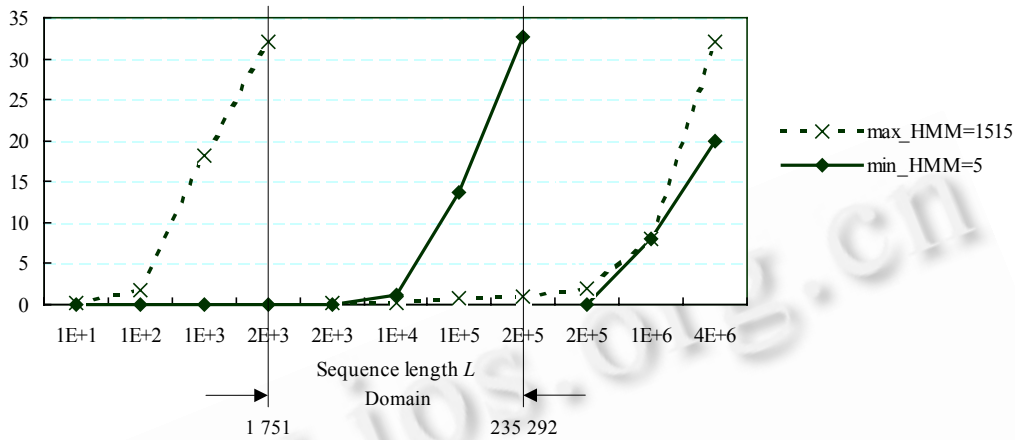


Fig.9 Viterbi memory requirements prediction when searching Pfam_ls database (whose maximum and minimum HMM model lengths are 1 515 and 5 respectively, and domain is the sequence length field in which the sequence may require the maximum memory when aligning with responding HMM model

图 9 Pfam_ls 上 viterbi 所需存储量估计(Pfam_ls 最大模型长度 1 515,最小为 5; Domain 为在不同模型长度上达到最大存储量的序列长度区域)

Table 6 The sequences investigation in two sequence files

表 6 真实序列文件中的序列落于 domain 区域的比重

Sequence file	Sequence length		Sequences whose length is in domain	
	Max.	Min.	Number	Percent (%)
vts3.fasta	37 419	12	966	0.9
200_seq.fasta	3 323	41	4	2

3 实验性能

由于所采用计算平台 PPC 的单节点内存容量满足 Pfam_ls 库上对序列进行比对内存的需求.因此,我们在 slave 一级采用“一次载入”HMM 结构的机制,即一次将该 slave 所需的 HMM 结构从硬盘读出,存储在该 slave 的缓冲区中,以后每次对该 HMM 的操作,则从该缓冲区中取出.同时,为了实现在缓冲区快速定位所需的 HMM 结构,我们采用了循环链表结构,记录下一时刻所要访问的链表位置.在 manager 一级仍然采用从硬盘中读取的方式,因为对于组内所有 slave 产生的有意义的比对结果,需要重新访问数据库,以获取附属信息.

采用同样的 200 条序列搜索 Pfam_ls 库,得到的并行性能如图 10 所示.图中在 p=64 和 100 时取 K=8,在 100<p<703 时取 K=16,在 p=703 时取 K=26.在 703 台处理机上达到 95%的效率,吞吐率由原来的 2.802 提高到 3.54.

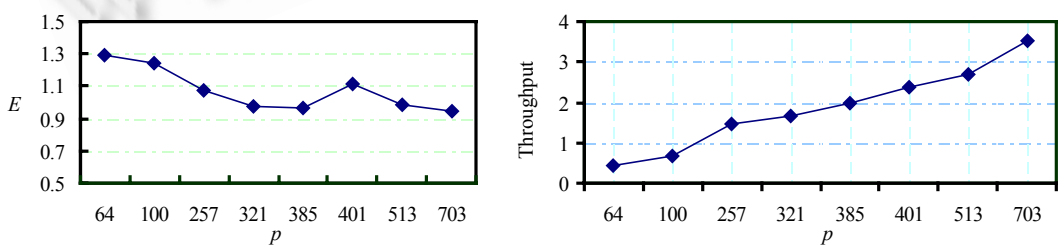


Fig.10 The final parallel performance of the optimized hmmpfam

图 10 优化后 Hmmpfam 的并行性能

出现超线性比是由于串行程序中 I/O 时间和计算时间是同量级的,当 p=64 时,slave 上 I/O 时间几乎是计算时间的 1/2.改进版本中极大地减少了 slave 访问硬盘的次数,若序列条数为 W,假设所有组的 slave 在分析完它处理的第 1 条序列后就做重分配负载工作,则 slave 上的 I/O 时间大约只有 I/O 优化前版本的 2/W 倍.对于多个序列,slave 只需一次把所需的 HMM 信息载入内存中,而无须重复载入,从而引起超线性比.

另外,在 p=401 处,并行效率有所上升,这是由于当 slave 数目增长到一定规模时,每个 slave 所要比对的 HMM 模型数目减少,所需 HMM 存储量减少,Cache 命中率提高所带来的好处大于通信开销的增加,从而使并行效率上升.但是,当 p 继续增大,对应的 slave 数目也继续增大时,通信开销的增加大于 Cache 命中率提高所带来的好处,并行效率开始下降.

4 结 语

生物计算是本世纪最富有活力的研究领域之一,而并行计算将加速该领域的发展.本文针对该领域广泛使用的 Hmmpfam 程序存在的不足,在并行机上对其算法进行了改进,并实现了 MPI 通用版本.该版本实现了序列和 HMM 模型两个层次的并行,采用了二级负载平衡策略,引入了 I/O 优化,在 703 台处理机上达到 95%的并行效率.该程序将被北京大学生物计算中心采用,实现二级基因数据库的快速搜索.

致谢 在此,我们向对本文的工作给予支持和帮助的老师和同学,尤其是北京大学生物计算中心罗静初教授、陈蕴佳、尹燕斌、高歌、国防科学技术大学计算机学院的周恩强、迟万庆、谢闵同志表示感谢.

References:

[1] Teresa KA, David JP. Luo JC, *et al*., Trans. Introduction to Bioinformatics. Beijing: Beijing University Press, 2002. 11~197 (in Chinese).

[2] http://www.genetics.wustl.edu/eddy/software/

[3] Eddy SR. Profile hidden Markov models. Bioinformatics, 1998,14(9):755~763.

[4] Richard D, Eddy SR, Anders K. Biological Sequence Analysis. Beijing: Tsinghua University Press, 2002. 46~79 (in Chinese).

[5] http://www.sgi.com/industries/sciences/chembio/htc.html

[6] http://www.apple.com/server/clustering-resource.html

[7] http://www.platform.com/PDFs/whitepapers/AC_Bioinformatics_WP_v3.pdf

[8] Hwang K, Xu ZW. Lu XD, *et al*., Trans. Scalable Parallel Computing Technology, Architecture, Programming. Beijing: China Machine Press, 2000. 416~458 (in Chinese).

[9] Mo ZY, Yuan GX. Message-Passing Parallel Programming Environment MPI. Beijing: Science Press, 2001. 1~11 (in Chinese).

附中文参考文献:

[1] Teresa KA, David JP,著.罗静初,等,译.生物信息学概论.北京:北京大学出版社,2002.11~197.

[4] Richard D, Eddy SR, Anders K.生物序列分析,蛋白质和核酸的概率论模型.北京:清华大学出版社,2002.46~79.

[8] 黄铠,徐志伟,著.陆鑫达,等,译.可扩展并行计算——技术、结构与编程.北京:机械工业出版社,2000.416~458.

[9] 莫则尧,袁国兴.消息传递并行编程环境 MPI.北京:科学出版社,2001.1~11.