

访存高效并行求解 (l, d) -Motif 发现问题

张 静¹, 钟 诚¹, 李 智^{1,2}

(1 广西大学 计算机与电子信息学院, 广西 南宁 530004; 2 广西科技信息网络中心, 广西 南宁 530012)

摘 要: 充分利用多核结构的共享三级缓存、私有二级和一级缓存, 将采用 Modeling 算法求解 (l, d) -Motif 发现问题过程中产生的大量不同位置组合的数据, 以多轮方式调度分配到各级缓存中, 通过线程绑定以平衡核心负载, 设计实现了一种访存高效的 (l, d) -Motif 发现线程级并行算法. 实验结果表明, 与已有的并行求解 (l, d) -Motif 发现算法相比, 本文给出的算法更高效、具有较好的加速比和可扩展性, 并且能够成功求解更长 l 的 (l, d) -Motif 发现问题.

关键词: (l, d) -Motif 发现; Modeling 算法; 组合计算; 多级缓存; 线程级并行

中图分类号: TP338.6; TP301.6

文献标识码: A

文章编号: 1000-7180(2014)06-0097-06

Cache-Efficient Parallel Solving (l, d) -Motif Finding Problem

ZHANG Jing¹, ZHONG Cheng¹, LI Zhi^{1,2}

(1 School of Computer and Electronics and Information, Guangxi University, Nanning 530004, China;

2 Guangxi Scientific Information Network Center, Nanning 530012, China)

Abstract: By utilizing fully the shared L3 Cache, private L2 Cache and L1 Cache on the multi-core architecture, the produced large-scale data with the different combination of positions are distributed to the multi-level caches while the Modeling algorithm is used to solve (l, d) -Motif finding problem, and a cache-efficient thread-level parallel (l, d) -Motif finding algorithm is designed on multi-core computer by binding the threads to the cores to balance their computational loads. The experimental results show that, compared with the existing parallel algorithm to solve (l, d) -Motif finding problem, the presented algorithm is efficient and has good speedup and scalability, and it can solve successfully the (l, d) -Motif finding problem with longer l .

Key words: (l, d) -Motif finding; Modeling algorithm; combinational computation; multi-level caches; thread-level parallelism

1 引言

Motif(模体)是指 DNA 序列中长度较短、具有保守功能的序列片段^[1], 从 DNA 序列中发现 Motif 的过程称为 Motif 发现, Motif 发现是生物信息学中一个基础而重要的研究问题^[1]. 植入 (l, d) -Motif 发现问题是指: 对于输入的 n 条 DNA 序列, 每条序列长度为 L 且都包含一个长度为 l ($l < L$) 的模体 M 的变体 M_i (称为模体实例), M_i 由 M 进化而来且 M 和 M_i 的汉明距离不大于 d , 在已知 l 和 d 的前提下,

从 DNA 序列中找出模体 M ^[1,2].

(l, d) -Motif 发现问题是一种组合优化、计算密集型的 NP hard 问题^[1], 当 l 较长时, 求解问题十分耗时. 已有的 Motif 发现算法主要分为基于字符串匹配的确定性算法和基于概率方法的启发式算法. 文献[3]中基于一阶马尔可夫链的分析方法提出一种在 DNA 序列中找出具有显著统计特性模式的 Wordup 算法. 文献[4]中将序列按照后缀树结构方式进行组织, 然后对后缀树形式做约减改进, 提出一种基于频繁子树挖掘的 DNA 重复序列识别方法.

收稿日期: 2013-08-22; **修回日期:** 2013-10-18

基金项目: 国家自然科学基金项目(60963001); 广西研究生教育创新计划项目(YCSZ2013006); 广西教育厅-广西大学博士点建设基金(P11900119)

文献[5]中提出的 PMSprune 算法成功解决了(17, 6)和(19, 7)-Motif 发现计算实例难题. 文献[6]中采用后缀树数据结构提出的 MITRA 算法解决了基于样本驱动的 SDA 算法^[7]在求解 (l, d) -Motif 发现问题时空间消耗过大的问题. 文献[8]中提出的投票算法给最多有 d 个位置突变的所有 d -变体集合中 l 的序列 s' 投票, 将 4^l 个长为 l 的输入序列分组, 然后逐个处理各组序列以求解 (l, d) -Motif 发现问题. 基于分群策略, 文献[2]中, 在同构机群系统上将序列分组分配给 p 个处理机结点并行处理, 然后归并并获得 (l, d) -Motif 发现问题的最终结果. 文献[9]中将序列后缀划分成若干组, 在单核处理器机群系统上实现了长序列最大串联重复识别并行算法. 文献[10]中运用二次采样方法改进 Gibbs 采样算法来求解 (l, d) -Motif 发现问题. 文献[11]中提出的 EM 算法要求每个序列中至少包含一个共同的位点, 交替执行 E 步(求期望值)和 M 步(计算最优值)直至收敛, 该算法求解过程依赖初始条件, 如果初始参数设置不当, 就不会得到最优结果. 文献[12]中提出一种聚类精炼算法来求解 (l, d) -Motif 发现问题, 这种方法自适应阈值以改变 Motif 实例的数量. 文献[13]中提出的 BitBased 算法运行 6.9 小时成功解决了(21, 8)-Motif 发现实例问题, 在此基础上, 该文利用 GPU 加速计算, 将求解(21, 8)-Motif 发现问题时间减少到 1.1 小时. Desraj 等人在文献[14]中提出的 Modeling 算法成功求解了(48, 12)-Motif 发现实例问题, 是目前已有公开报道文献中解决 l 较长的 (l, d) -Motif 发现算法; 在此基础上, 他们将算法并行化使之在多处处理器平台上运行, 取得良好的加速效果^[15].

本文在分析求解 (l, d) -Motif 发现问题的 Modeling 算法基础上, 充分利用多核结构的多级缓存, 将算法执行过程中产生的不同位置组合数据以多轮方式调度分配到各级缓存中, 以使得算法在并行求解 (l, d) -Motif 发现问题中实现访存高效; 同时通过线程绑定方法实现核心负载平衡、减少线程迁移所需的时间开销, 在多核计算机上实现了并行求解更长 l 的加速比高、可扩展性好的 (l, d) -Motif 发现算法.

2 Modeling 算法的分析

任意一条长为 l 的生物序列记为 l -mer; 对于任意两条 l -mer 序列, 若相同位置上对应的字符不同, 则字符不同的位置总数记为两序列的“汉明距离”.

定义 1 Modeling 计算^[14]: 对于两条汉明距离

为 $2d$ 的 l -mer 序列 ls_1 和 ls_2 , 记录 ls_1 和 ls_2 的 $2d$ 个不同位置; 从 ls_1 的 $2d$ 个位置中任意选择 d 个位置 (共有 $r = \binom{2d}{d}$ 种选择情形), 将这 d 个位置的字符替换为 ls_2 中相应位置的字符, 得到结果序列 s . 计算序列 s 的过程称为 Modeling 计算.

设 ls_1 和 ls_2 的 $2d$ 个不同位置分别为 $T_1, T_2, \dots, T_{2d}$, 从此 $2d$ 个位置中任意选择 d 个位置的不同位置组合情形共 $\binom{2d}{d} = \binom{2d-1}{d-1} + \binom{2d-2}{d-1} + \dots + 1$ 个, 其中 $\binom{2d-1}{d-1}$ 为以 T_1 开始的位置组合情形数, $\binom{2d-2}{d-1}$ 为以 T_2 开始的位置组合情形数, $\dots, 1$ 为以 T_{d+1} 开始的位置组合情形数, 则全部的组合情形可表示如下:

第 1 种组合情形: $(T_1, T_2, \dots, T_{d-1}, T_d)$,

第 2 种组合情形: $(T_1, T_2, \dots, T_{d-1}, T_{d+1})$,

...

第 $\binom{2d-1}{d-1}$ 种组合情形: $(T_1, T_{d+2}, \dots, T_{2d-1}, T_{2d})$,

...

第 $r - 1$ 种组合情形: $(T_d, T_{d+2}, \dots, T_{2d-1}, T_{2d})$,

第 r 种组合情形: $(T_{d+1}, T_{d+2}, \dots, T_{2d-1}, T_{2d})$.

设有 n 条长度分别为 L 的 DNA 序列, BS_i 为第 i 条 DNA 序列, R_{ij} 表示序列 i 中起始位置为 j 的长为 l 的子序列, C 表示与 R_{ij} 汉明距离小于等于 $2d$ 的长为 l 的序列集合, CM 表示与 R_{ij} 汉明距离等于 $2d$ 的长为 l 的序列集合, $i = 1 \sim n, 1 \leq j \leq L - l + 1$, c_k 表示 CM 集合中第 k 条序列, $k = 1 \sim |CM|$. 利用 Modeling 计算方法, 文献[14]中提出的求解 (l, d) -Motif 发现问题的串行算法如下:

算法 1 (l, d) -Motif 发现 Modeling 算法^[14]

输入: $BS_i (i = 1 \sim n), L, l, d$

输出: $M(\text{Motif})$

Begin

(1) for $i = 1$ to n do

(2) for $j = 1$ to $L - l + 1$ do

(2.1) 检测 R_{ij} 是否为 Motif, 若是则输出 Motif, 算法结束;

(2.2) 构造 R_{ij} 对应的集合 C 和 CM ;

(2.3) for $k = 1$ to $|CM|$ do

(2.3.1) 记录 R_{ij} 与 c_k 之间 $2d$ 个不同的位置及从此 $2d$

个位置中任意选取 d 个位置的 $r = \binom{2d}{d}$ 个位置组合;

(2.3.2) for $t = 1$ to r do

(2.3.2.1) 对 R_{ij} 和 c_k 不同位置组合情形进行 Modeling 计算, 得到结果 R_{ij}' ;

(2.3.2.2) 检测 R_{ij}' 是否为 Motif, 若是, 则输出 Motif, 算法结束;

End

文献[14]的研究结果表明, 对不同长度 l 和 d 的 (l, d) -Motif 发现问题, 当 $d \leq 1/4$ 时 Modeling 算法表现出较好的运行性能, 与已有的 (l, d) -Motif 发现算法相比, Modeling 算法能有效解决 $l=48$ 的 (l, d) -Motif 发现问题, 成功率也比较高。

算法1中步骤(2.1)所需时间为 $O(n)$, 步骤(2.2)所需时间为 $O(n)$, 步骤(2.3.1)所需时间为 $O(D)$, 步骤(2.3.2.1)所需时间为 $O(d)$, 步骤(2.3.2.2)所需时间为 $O(|C|)$. 因此, 算法1时间复杂度为 $O(n \times (L-l+1) \times (2n+|CM| \times (l+r \times (d+|C|))))$. 由此可以看出其计算时间主要取决于 $|C|$ 和 r 的大小, 随着 l 和 d 的增大, $|C|$ 逐渐减小, 但 r 值急剧增大, 这表明 Modeling 算法需要处理的组合情形的数据非常多, 十分耗时。

为此, 本文研究提出高效、可扩展、多线程并行求解 (l, d) -Motif 发现问题算法, 以发现 l 更长的模板 M , 更好地满足生物信息学研究的需求。

3 基于 Modeling 计算的 (l, d) -Motif 发现并行算法

3.1 多级缓存数据划分方法

随着 l 和 d 的增大, r 值急剧增加, 特别地, 当 $l > 44, d > 11$ 时, Modeling 计算需要处理的组合情形达到上百万甚至更多, 这使得算法在频繁处理各种组合情形的替换和匹配比对操作时, 所需处理数据时常不在缓存中, 造成 Cache 缺失, 算法在执行过程中需要在主存和缓存之间多次数据交换, 导致算法效率低下. 因此, 为了减少多核并行算法求解 (l, d) -Motif 发现问题所需的访存开销, 我们需要研究对计算过程中产生的各种位置组合的数据进行适当划分并调度分配存储到多核机器的三级、二级和一级缓存中, 以使得各核心在计算过程中所需数据能很快获得, 实现访存高效. 数据划分方法如下:

(1) 主存到共享三级缓存的数据划分

将主存中计算 (l, d) -Motif 发现问题的各种位置组合的规模为 $r \times d$ 的数据逐轮分配到共享三级缓存中, 每轮分配规模为 $\alpha \times C_3$ 的数据, 总共进行 d_3

$= \frac{r \times d}{\alpha \times C_3}$ 轮从主存到三级缓存的数据分配, 其中 C_3

为三级缓存可容纳的整数个数, α 为三级缓存利用率, $\alpha \leq 1$.

(2) 共享三级缓存到私有二级缓存的数据划分

将共享三级缓存中规模为 $\alpha \times C_3$ 的数据逐轮分配到每个核心私有的二级缓存中, 每轮分配规模为

$\beta \times C_2 \times p$ 的数据, 总共进行 $d_2 = \frac{\alpha \times C_3}{\beta \times C_2 \times p}$ 轮从三

级缓存到二级缓存的数据分配, 其中 C_2 为二级缓存容量可容纳的整数个数, p 为多核机器的核心数, β 为二级缓存利用率, $\beta \leq 1$.

(3) 二级缓存到一级缓存的数据划分

将每个核心的二级缓存中规模为 $\beta \times C_2$ 的数据逐轮分配到其一级缓存中, 每轮分配规模为 $\gamma \times C_1$

的数据, 总共进行 $d_1 = \frac{\beta \times C_2}{\gamma \times C_1}$ 轮从二级缓存到一级

缓存的数据分配, 其中 C_1 为一级缓存容量可容纳的整数个数, γ 为一级缓存利用率, $\gamma \leq 1$.

3.2 多级缓存数据划分的 (l, d) -Motif 发现并行算法

多级缓存数据划分的多线程并行求解 (l, d) -Motif 发现问题的过程如图1所示。

在通过多级缓存划分各种位置组合数据以实现访存高效并行求解 (l, d) -Motif 发现问题的基础上, 为了使得各个核心负载均衡, 避免线程在核心之间迁移产生额外的时间开销, 提高算法的运行性能, 本文利用线程亲和力函数 `sched_setaffinity()` 将线程号为 t 的线程绑定到编号为 $t \bmod p$ 的核心上运行。

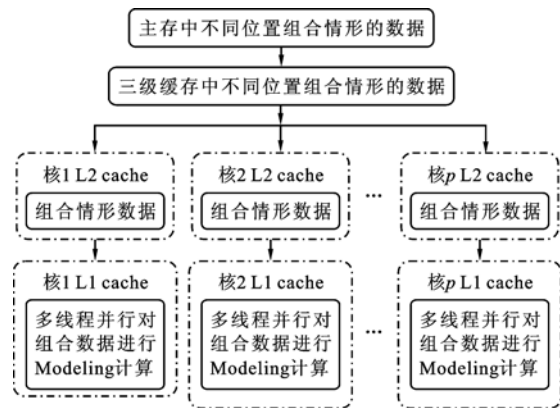


图1 多线程并行求解 (l, d) -Motif 发现问题的过程

多核系统上基于多级缓存数据划分求解 (l, d) -Motif 发现问题的线程级并行算法描述如下:

算法2 基于数据划分的 (l, d) -Motif 发现线程级并行算法

输入: $BS_i (i=1 \sim n), L, l, d$

输出: $M(\text{Motif})$

Begin

(1) 计算从主存到三级缓存的调度分配轮数 d_3 , 每轮分配的数据规模为 $\alpha \times C_3$;

(2) 计算从三级缓存到二级缓存的调度分配轮数 d_2 , 每轮分配的数据规模为 $\beta \times C_2 \times p$;

(3) 计算从二级缓存到一级缓存的调度分配轮数 d_1 , 每轮分配的数据规模为 $\gamma \times C_1$;

(4) for $i = 1$ to n do

(5) for $j = 1$ to $L - l + 1$ do

(5.1) 对于 BS_i 之外的其他 $n-1$ 条 DNA 序列, 若每条序列中至少有一条长为 l 的子序列与 R_{ij} 之间的汉明距离小于等于 d , 则 R_{ij} 为所要找的 Motif, 输出 Motif, 算法结束;

(5.2) 构造 R_{ij} 对应的集合 C 和 CM ;

(5.3) for $k = 1$ to $|CM|$ do

(5.3.1) 记录 R_{ij} 与 c_k 之间 $2d$ 个不同的位置及从此 $2d$ 个位置中任意选取 d 个位置的 $r = \begin{Bmatrix} 2d \\ d \end{Bmatrix}$ 个位置组合;

(5.3.2) for $t_3 = 1$ to d_3 do

{将主存规模为 $\alpha \times C_3$ 的数据读入到 L3 cache;

for $t_2 = 1$ to d_2 do

{将 L3 cache 中规模为 $\beta \times C_2 \times p$ 的数据读入到 L2 cache;

for $t_1 = 1$ to d_1 do

{将 L2 cache 中规模为 $\gamma \times C_1$ 的数据读入到 L1 cache;

for $th = 1$ to t_s do in parallel

// t_s 为并行线程数

for $t_0 = 1$ to c_1/t_s do

对 R_{ij} 和 c_k 不同位置组合情形进行 Modeling 计算, 得到结果 R_{ij}' ; 若 R_{ij}' 为 Motif, 则输出 Motif, 算法结束;

}}}

End

算法 2 中步骤 (5.1) 所需时间为 $O(n)$, 步骤 (5.2) 所需时间为 $O(n)$, 步骤 (5.3.1) 所需时间为 $O(d)$, 步骤 (5.3.2) 所需时间为 $O(d_3 \times d_2 \times d_1 \times c_1/t_s \times (d + |C|))$.

因此, 算法 2 的时间复杂度为 $O(n \times (L - l + 1) \times (2n + |CM| \times (l + d_3 \times d_2 \times d_1 \times c_1/t_s \times (d + |C|))))$, 加速度为

$$O\left(\frac{2n + |CM| \times (l + r \times (d + |C|))}{2n + |CM| \times (l + d_3 \times d_2 \times d_1 \times c_1/t_s \times (d + |C|))}\right).$$

4 实验

4.1 实验环境与数据

实验的硬件平台是拥有 2 个 Intel XEON E5620 2.4 GHz 处理器、每个处理器有 4 个核心的

多核计算机, 其主存容量为 12 GB, 共享三级缓存容量为 12 MB, 每个核心私有的二级缓存容量为 256 kB, 一级缓存容量为 64 kB, 运行的操作系统为 Red Hat Enterprise Linux 5, 采用 C 语言和 OpenMP 编程实现本文提出的求解 (l, d) -Motif 发现问题的多核线程级并行算法, 并与文献[15]的算法进行测试对比。

实验数据: 模拟生成 20 条长度分别为 600 的 DNA 序列及一条长为 l 的 Motif 序列, 序列中每个位置的碱基 (A, T, C, G) 等概率地独立生成, 然后在这 20 条序列中的随机位置将 Motif 的变体植入。

4.2 实验结果与分析

采用动态关闭核心的方法, 在运行不同数目的核心、不同数目的线程、不同长度的 l 和 d 时, 测试本文算法并行求解 (l, d) -Motif 发现问题所需的时间、获得的加速比和可扩展性, 并与文献[15]的 (l, d) -Motif 发现并行算法进行实验性能对比。

图 2~图 5 分别给出了本文算法在运行 8、6、4 和 2 个核心以及不同数目的线程并行求解不同长度的 l 和 d 的 (l, d) -Motif 发现问题所需的运行时间。

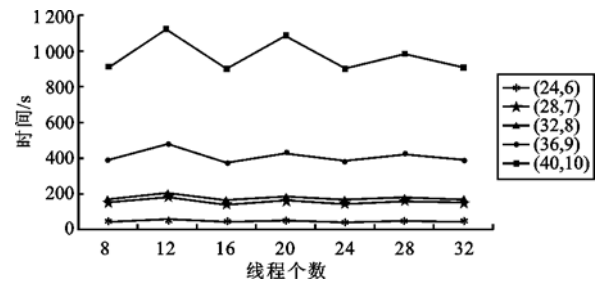


图 2 运行 8 个核心和不同数目的线程, 本文算法并行求解不同规模 (l, d) -Motif 发现问题所需的时间

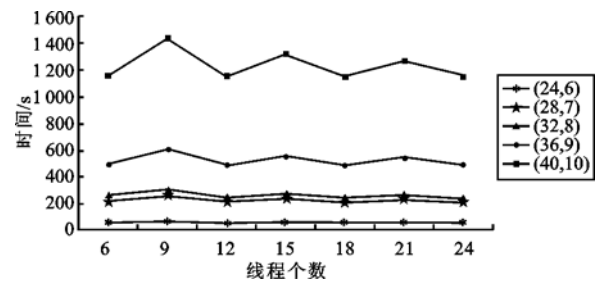


图 3 运行 6 个核心和不同数目的线程, 本文算法并行求解不同规模 (l, d) -Motif 发现问题所需的时间

由图 2~5 结果可以看出, 对于不同长度的 l 和 d , 算法运行的核心越多, 其求解 (l, d) -Motif 发现问题所需的时间越少; 同时可以看出, 当算法运行的线程数是运行的核心数的整数倍时, 它所需的时间比运行的线程数不是运行的核心数的整数倍时所需时间少, 这是因为当运行的线程数是运行的核心数的整数倍时, 每个线程都能绑定到一个核心上运行,

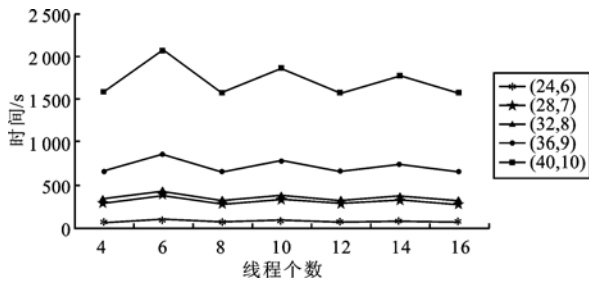


图4 运行4个核心和不同数目的线程,本文算法并行求解不同规模 (l, d) -Motif发现问题所需的时间

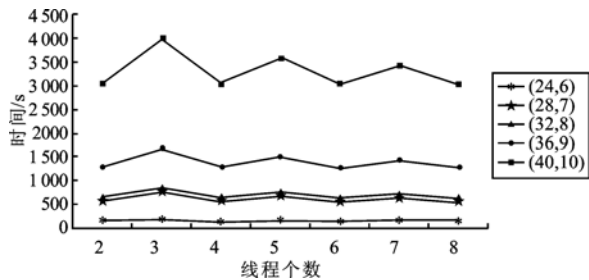


图5 运行2个核心和不同数目的线程,本文算法并行求解不同规模 (l, d) -Motif发现问题所需的时间

减少了线程在核心之间的来回迁移,也能保持每个核心之间的负载均衡.此外,由图2~5结果还可以看出,当运行的线程数是运行的核心数的2或3倍时,算法所需的时间最少,我们称这样的线程数为核心运行的最佳线程数.

图6给出了每个核心运行最佳数目的线程时,本文算法并行求解不同长度的 l 和 d 的 (l, d) -Motif发现问题获得的加速比.

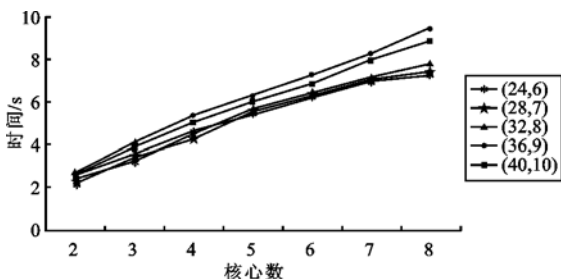


图6 各核心运行最佳数目的线程时,本文算法并行求解不同规模 (l, d) -Motif发现问题的加速比

由图6的结果可以看出,对于不同规模的 (l, d) -Motif发现问题,随着运行核心的增加,本文算法获得的加速比也相应增加.

对于不同规模的 (l, d) -Motif发现问题(l 增加至48),图7中给出了本文算法和文献[15]算法在运行8个核心及相应最佳数目的线程时所需的运行时间.

图7的实验结果表明,当 $l < 36$ 时,本文算法和文献[15]算法所需的运行时间几乎一样;但当 l 逐渐增大时,本文算法所需的时间明显低于文献[15]

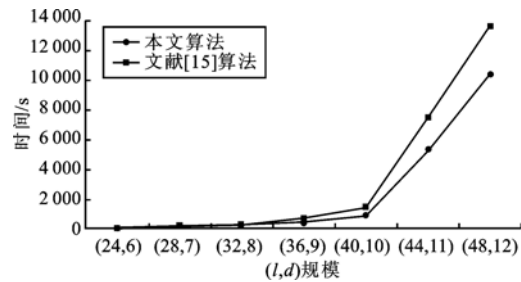


图7 本文算法和文献[15]算法并行求解不同规模 (l, d) -Motif发现问题所需的时间

算法所需的时间,这是因为当 l 较大时,算法在执行过程中产生的组合情形数据很多,本文算法利用多级缓存划分数据方法,依据各级缓存容量大小,将主存中的组合数据以多轮方式调度分配到三级、二级和一级缓存中,使得各核心在并行求解 (l, d) -Motif发现问题的过程中,减少了数据在主存和缓存之间的交换次数,从而明显减少访存时间,这使得本文算法在处理较长 l 的 (l, d) -Motif发现问题时表现出较好的性能.

按照等速度度量标准^[16,17],我们测试了本文算法的可扩展性.文献[16,17]定义的并行计算的速度为 $V=W/T$,其中 W 为工作量或问题规模(本文算法使用组合情形数据作为其工作量), T 为并行计算时间.图8中给出了随着运行的核心逐步增多时,本文算法并行求解规模分别为 $(36, 9)$ 、 $(40, 10)$ 和 $(48, 12)$ 的Motif发现问题的速度曲线.

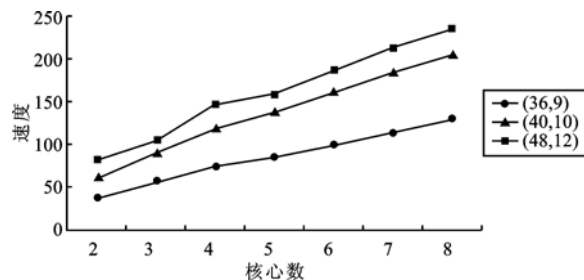


图8 运行核心逐步增多时,本文算法并行求解不同规模的 (l, d) -Motif发现问题的速度曲线

从图8可以看出,本文算法并行求解不同规模的 (l, d) -Motif发现问题的速度曲线随着运行的核心增加呈近似线性增加,这说明本文算法具有较好的可扩展性.

此外,为了测试本文算法可否并行求解更长 l 的 (l, d) -Motif发现问题实例,我们运行13小时成功求解了 $(52, 13)$ -Motif发现问题.

5 结束语

针对 l 较长时组合情形数据很多的 (l, d) -Motif发现问题,本文提出一种多级缓存数据划分方法,将

组合情形数据以多轮方式调度分配到各级缓存中,使得各个核心并行求解 (l, d) -Motif 发现问题时,大大减少了数据在主存和缓存之间的交换次数,实现了并行算法的访存高效;同时,通过线程绑定平衡了各计算核心的负载、减少了线程迁移开销. 本文算法具有良好的可扩展性,在并行求解较长 l 的 (l, d) -Motif 发现问题时,其所需的运行时间明显少于同类研究的其他算法.

为了能够在可接受的有限时间内求解更长 l 和 d 的 (l, d) -Motif 发现问题,下一步的工作将研究设计多核 CPU/GPU 混合结构机群上 CPU 和 GPU 协同求解 Motif 发现问题的并行算法.

参考文献:

- [1] Pevzner P, Sze S. Combinatorial Approaches to Finding Subtle Signals in DNA Sequences[C] // Proc of the Eighth International Conference on Intelligent Systems on Molecular Biology, San Diego, CA: AAAI Press, 2000: 69-278.
- [2] 李锦, 钟诚, 卿海军. 基于分群策略的机群系统上 Motif 发现并行算法[J]. 计算机工程与科学, 2008, 30(11): 126-128.
- [3] Pesole G, Prunella N, Liuni S, et al. WORDUP: an efficient algorithm for discovering statistically significant patterns in DNA sequences[J]. Nucleic Acids Res, 1992, 30(11): 2871-2875.
- [4] 周溜溜, 业宁, 徐昇, 等. 基于频繁子树挖掘的 DNA 重复序列识别方法[J]. 微电子学与计算机, 2011, 28(9): 193-196.
- [5] Davila J, Balla S, Rajasekaran S. Fast and practical algorithms for planted (l, d) motif search[J]. IEEE/ACM Trans. on Computational Biology and Bioinformatics, 2007, 4(4): 544-552.
- [6] Eskin E, Pevz P. Finding composite regulatory patterns in DNA sequences[J]. Bioinformatics, 2002, 18(1): 354-363.
- [7] Waterman M S, Arratia R, Galas D J. Pattern recognition in several sequences: consensus and alignment[J]. Bull Math Biol, 1984(46): 515-527.
- [8] Chin F, Leung H. Voting Algorithms for Discovering Long Motifs[C] // Proc of 2005 Asia-Pacific Bioinformatics Conference, London: Imperial College Press, 2005: 261-271.
- [9] 陆向艳, 钟诚. 机群系统上长序列最大串联重复识别并行算法[J]. 微电子学与计算机, 2010, 27(8): 186-189.
- [10] 刘佳. 基于 Gibbs 算法的改进算法及其在识别 MOTIF 中的应用[D]. 长春: 吉林大学, 2010.
- [11] Lawrence C E, Reilly A A. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences [J]. Proteins, 1990, 7(1): 41-51.
- [12] Li Gang, Chan Tak-Ming, Leung Kwong-Sak, et al. A cluster refinement algorithm for motif discovery[J]. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2010, 7(4): 654-667.
- [13] Dasari N S, Desh R, Zubair M. An efficient multicore implementation of planted motif problem [C] // Proc of 2010 International Conference on High Performance Computing and Simulation. France: IEEE, 2010: 9-15.
- [14] Desaraju S, Mukkamala R. Solving planted motif problem using modeling method[C] // Proc 2011 International Conference on Bioinformatics & Computational Biology. Las Vegas, USA: IEEE, 2011: 1-7.
- [15] Desaraju S, Mukkamala R. Multiprocessor implementation of modeling method for planted motif problem [C] // Proc of 2011 World Congress on Information and Communication Technologies. Mumbai: IEEE, 2011: 524-529.
- [16] Sun X H, Rover D T. Scalability of parallel algorithm-machine combinations [J]. IEEE Trans on Parallel and Distributed Systems, 1994, 5(6): 519-613.
- [17] 陈国良. 并行计算机体系结构[M]. 北京: 高等教育出版社, 2002.

作者简介:



张 静 女, (1988-), 硕士研究生. 研究方向为网络与并行分布计算.



钟 诚 男, (1964-), 博士, 教授, 博士生导师. 研究方向为网络与并行分布计算、计算生物学.

李 智 男, (1967-), 硕士, 研究员. 研究方向为网络与并行分布计算.