B-RPL:低存储开销的RPL路由协议

杨 红[1,2,3] 朱红松[1,3] 孙利民[1,3]
(中国科学院信息工程研究所信息安全国家重点实验室 北京 100093)[1]
(中国科学院大学 北京 100049)[2] (物联网信息安全技术北京市重点实验室 北京 100093)[3]

摘 要 针对低功耗易失网络(Low-power and Lossy Networks,LLNs)中存储式RPL路由的大存储开销问题,提出了一种基于存储式RPL的改进型路由协议B-RPL。该协议充分利用了LLNs网络无线通信的广播特性,将RPL中的路由表简化为目的节点集合,并利用布隆过滤器(Bloom Filter)管理该目的节点集合,极大地减少了节点的存储开销。此外,B-RPL还包含了一些针对网络拓扑动态变化的自适应机制。实验及分析表明:与存储式RPL相比,B-RPL节约了97.8%的存储开销,而通信开销仅增加2.4%。

关键词 物联网,6LoWPAN,RPL,路由,布隆过滤器
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.023

B-RPL:Low Memory Cost RPL Routing Protocol

YANG Hong[1,2,3] ZHU Hong-song[1,3] SUN Li-min[1,3]
(State Key Laboratory of Information Security,Institute of Information Engineering,Chinese Academy of Sciences,Beijing 100093,China)[1]
(University of Chinese Academy of Sciences,Beijing 100049,China)[2]
(Beijing Key Laboratory of IOT Information Security Technology,Institute of Information Engineering,CAS,Beijing 100093,China)[3]

Abstract The RPL routing protocol is widely used in low-power and lossy networks (LLNs). However,the storing mode of RPL is criticized for large memory consumption. In this paper,an advanced RPL was proposed,called B-RPL,which reduces the memory cost by making routing decision according to a set of destinations rather than the routing table used in the raw RPL. Further more,the B-RPL employs Bloom filter to manage the set of destinations,so that the memory consumption becomes extremely low. The B-RPL also contains several adaptive designs specially tailored for dynamic network change in LLNs. Experiments show that,comparing with RPL in storing mode,B-RPL saves 97.8% storage at the expense,and only increases 2.4% transmission overhead.

Keywords Internet of things,6LoWPAN,RPL,Routing,Bloom filter

1 引言

近年来,随着技术和应用的发展,物联网(Internet of Things)已经广泛渗入到电力、交通、医疗、家居、工业 、农业、物流、军事、环境、节能减排、公共安全等行业。物联网强调前端智能化和网络化,在前端网络中存在大量的微型感知设备和控制器。这些微型设备的处理和存储能力都非常有限,它们通过低功耗的无线网络(例如IEEE802.15.4)连接在一起,并通过网关间接接入Internet。为了使物联网中每个微型设备都能通过统一的协议与Internet连接,IETF(Internet Engineering Task Force)任务组成立了6LoWPAN0工作组。其目的是将IPv6引入到物联网的前端网络。

为了解决6LoWPAN中的路由问题,IETF于2008年成立了ROLL(Routing over Low-power and Lossy Networks)工作组。该工作组提出了RPL(Routing Protocol for LLNs)协议[2]。虽然该协议目前还是工作组草案,但是它已经被应用到多种物联网系统中,同时也引起学术界的广泛关注[3-5]。RPL协议支持3种类型的数据通信模型[6],即普通节点到根节点(sink节点、汇聚节点)的多点到点的通信MP2P(Multi-Point-To-Point),根节点到多个普通节点的点到多点通信P2MP(Point-To-MultiPoint)以及普通节点之间点到点的通信P2P(Point-To-Point)。

RPL通过建立树形的面向目的节点(即根节点)的有向无环图DODAG(Destination Oriented Directed Acyclic Graph)来形成整个网络的路由拓扑图,并在此基础上提供两种路由工作模式:非存储模式和存储模式。在非存储模式中,只有根节点存储所有网络拓扑信息,完整的路由信息被存储在数据包中一起发向目的节点。在存储模式中,根节点和普通节点需要维护一个路由表,用于存储其每个子孙节点的下一跳信息。数据包中只需要包含目的节点的地址,而不需要完整的路由

到稿日期:2014-01-13 返修日期:2014-04-25 本文受国家高技术研究发展计划(863)(2012AA050804),北京市科委科技创新基地培育与发展工程专项项目(Z131101002813085),中国科学院信息工程研究所前瞻部署项目(Y3Z0071G02)资助。

杨 红(1987-),男,硕士生,主要研究方向为物联网技术、无线传感器网络,E-mail:yanghong@iie.ac.cn;朱红松(1973-),男,博士,副研究员,主要研究方向为无线通信、无线传感器网络、虚拟技术、物联网大数据安全和智能处理;孙利民(1966-),男,博士,研究员,主要研究方向为无线传感器网络、物联网安全、物联网隐私保护。

信息。

大量的研究工作(如文献[7-11])分析了非存储模式和存储模式下 RPL 的优缺点。非存储模式 RPL 的优点是普通节点的存储开销小,但传输过程中的通信开销大;存储模式恰好与之相反,即普通节点的存储开销大,但传输过程的通信开销小[9,10]。甘伟等人[7]提出了一种结合非存储模式和存储模式的优化方法,即普通节点也存储一部分网络拓扑信息,使得在数据包中不需要包含完整的路由信息,达到减少通信开销的目的。本文针对存储模式中普通节点存储开销大的问题,利用物联网中低功耗网络无线通信的广播特点和布隆过滤器(Bloom Filter)技术,提出 B-RPL 协议。实验及分析表明,B-RPL 能够极大地减少普通节点的存储开销。

2 RPL 简介

2.1 DODAG 图的建立

RPL 是针对低功耗易失网络 LLNs 制定的基于 IPv6 的距离矢量路由协议。它沿着一个以汇聚节点或者网关节点为根节点的有向无环图 DODAG 进行路由选择,如图 1 所示。根节点通过周期性地广播 DODAG 信息对象帧(DODAG Information Object,DIO)来维护 DODAG 图。DIO 帧中包含根节点编号、路由指标和发送节点的路由深度("rank")。接收节点根据 DIO 帧的信息计算自己的路由深度,并选择能够使自己路由深度最小的 DIO 帧的发送节点作为父节点。然后,该接收节点广播出自己的 DIO 帧。

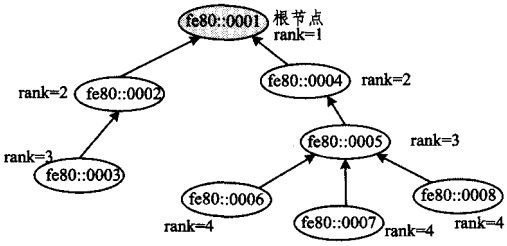

图 1 由 8 个节点组成的网络的 DODAG 图

选定父节点之后,每个节点需要发送一个目的通告对象帧(Destination Advertisement Object,DAO)给父节点,以便通知所有祖先节点自己已经加入该网络。非存储和存储模式中 DAO 帧的转发方式不相同。在非存储模式中,父节点直接转发子孙节点的 DAO 帧,根节点接收到所有节点的 DAO 帧,并根据所有的 DAO 帧计算出网络的 DODAG 图。在存储模式中,父节点在转发子孙节点的 DAO 帧的同时,维护一种如图 2 所示的路由表,用于记录到达各个子孙节点路由的下一跳。

目的节点 128bit	下一跳 128bit
fe80::0006	fe80::0005
fe80::0008	fe80::0005

图 2 存储式 RPL 中节点存储的路由表

2.2 路由过程

任意节点一旦选定了父节点,该节点就有了一条到根节点的 MP2P 路由。节点将数据包发送给父节点,父节点再转发给自己的父节点,如此反复,直至传输到根节点。

P2MP 路由根据 RPL 是非存储模式或者存储模式而有所不同。非存储模式中,根节点根据 DODAG 图计算出到达目的节点的路由,并把完整的传输路径信息随着数据包一起发向目的节点。中间的节点接收到数据包之后,根据数据包中的传输路径信息直接转发该数据包。存储模式中,数据包中的路由信息只有源和目的节点的编号,根节点和中间节点根据各自的路由表信息计算出下一跳节点,再进行数据包转发。

P2P 路由在非存储和存储模式下也有所不同。非存储模式中,源节点需要先将数据包传输至根节点,再由根节点利用 P2MP 路由将数据包转发至目的节点。而存储模式中,源节点沿着 MP2P 路由将数据包发送至目的节点的公共祖先节点时,该公共祖先节点根据存储的路由表信息将数据包转发至目的节点。

3 B-RPL

从第 2 节的讨论可以得知,RPL 路由协议的存储模式中 P2MP 和 P2P 路由的传输开销非常低。但每个节点的存储开销非常大,尤其是靠近根节点的节点。本节提出的 B-RPL 路由协议结合了无线通信的广播特性和布隆过滤器(Bloom Filter)[12]的存储高效性,极大地减少了存储式 RPL 的存储开销。

3.1 协议框架

B-RPL 协议在存储式 RPL 路由协议的基础上进行了两方面的修改:转发方式和路由表结构。

B-RPL 中,节点采用广播的方式转发数据包。LLNs 网络主要采用低功耗无线通信协议 ZigBee[13](IEEE 802.15.4)。该协议的调制速率固定(250kbps),且采用单天线通信。这使得这类网络的无线通信具有良好的广播特性。也就是说,对于给定的发送节点和接收节点,无论发送节点是将数据包单播发送给接收节点,还是将数据包广播出去,接收节点成功接收该数据包的概率相同,且两种情况下的吞吐率也相同。值得注意的是,对于 IEEE 802.11 中的通信协议,虽然它们的通信也具有一定的广播特性,但由于多调制速率和多天线的特点,其单播的吞吐率要高于广播。

B-RPL 的节点只接收处理其父节点广播出来的数据包,并进行路由决策,决定是否继续广播该数据包。B-RPL 中的路由决策不再使用 RPL 中的路由表,而是利用一个目的地址集合。该集合存储了所有需要本节点进行转发的目的地址。为了提高存储效率,B-RPL 利用了布隆过滤器来维护目的地址集合。后面几个小节将进行详细介绍。

3.2 Bloom Filter 机理

布隆过滤器(Bloom Filter)是一个存储空间高效的概率性数据结构。它用于测试一个元素是否存在于某个集合之中,测试结果存在一定的误报率(false positive),但不存在漏报(false negative)。如果布隆过滤器的测试结果显示某个元素存在于集合之中,则该元素以较高的概率存在于集合中。如果布隆过滤器的测试结果为某个元素不存在于集合中,则该元素一定不存在于集合中。

一个空布隆过滤器是一个长为 m 的全 0 位(bit)数组。另外它包括了 k 个哈希(hash)函数,每个哈希函数能够独立随机地将一个元素映射到位数组的某个固定位置。当向布隆

过滤器中添加元素的时候，只需要计算出该元素经过 k 个哈希函数映射得到的 k 个位置，然后将位数组的这 k 个位置上的值设置为 1。查询一个元素是否在布隆过滤器中时，同样计算出该元素经过 k 个哈希函数映射得到的 k 个位置。如果位数组中的这 k 个位置的值都为 1，则布隆过滤器报告该元素被插入过，即存在于对应的集合中；反之，如果位数组中的这 k 个位置的值有一个为 0，则布隆过滤器报告该元素未被插入过，即不存在于对应的集合中。

当布隆过滤器中的 k 个哈希函数都能够以均等概率选择位数组的每个位置时，文献[12]给出了布隆过滤器的误报率的近似表达式：

$$P\approx(1-e^{-kn/m})^k \tag{1}$$

其中，n 是集合中当前已经插入的元素个数。文献[12]还指出，对于给定的 m 和 n，使得误报率最小的 k 为：

$$k=\frac{m}{n}\ln 2 \tag{2}$$

由式(1)和式(2)可得：

$$P\approx 2^{-k}=2^{-\frac{m}{n}\ln 2} \tag{3}$$

3.3 协议设计

本小节详细描述 B-RPL 基于布隆过滤器的地址集合的建立、P2MP 和 P2P 通信以及协议中的一些关键技术。MP2P 通信在 B-RPL 和 RPL 中完全相同，因此不再介绍。

3.3.1 协议过程

根据 2.1 小节介绍的内容可知，节点处理完 DIO 帧并选定了父节点之后，将发送一个 DAO 帧通知父节点。在 B-RPL 中，每个节点收到子孙节点的 DAO 帧后，将该子孙节点的 IP 地址添加进布隆过滤器，并转发该 DAO 帧给自己的父节点。

P2MP 通信时，每个节点接收父节点广播出来的数据包，并检查数据包的目的地址是否在自己的布隆过滤器中。如果不在则丢弃该数据包；反之则转发该数据包，即把该数据包继续广播出去。

P2P 通信时，B-RPL 不能像存储式 RPL 那样，即只将数据转发至源节点和目的节点的最近公共祖先。这是因为布隆过滤器存在误报率，导致 B-RPL 无法正确地判断出源节点和目的节点的公共祖先。但网络中的所有节点都有一个公共祖先，即根节点。因此 B-RPL 的 P2P 通信中，每个节点先利用 MP2P 通信将数据包发送至根节点，再由根节点利用 P2MP 通信将数据包发送至目的节点。

与 RPL 相同，B-RPL 中的根节点会周期性地广播 DIO 包来维护 DODAG 图。每轮维护 DODAG 图时，所有节点清空之前的布隆过滤器，重新向布隆过滤器添加子孙节点。

3.3.2 性能分析

B-RPL 中，每个节点的存储开销为该节点的布隆过滤器中位数组的长度 m。而每个数据包中的通信开销等于存储式 RPL 路由，即只含有源和目的地址，不必像非存储式 RPL 那样在数据包中包含完整的传输路径信息。但由于布隆过滤器具有一定的误报率，将导致中间节点可能进行一些不必要的数据通信，进而引入额外的通信开销。

假设 DODAG 是一个均衡 M 叉树，即除叶子节点之外，每个节点都有 M 个子节点，且任意两叶子节点的深度之差不大于 1。假设 DODAG 图的深度为 R。当根节点发送一个数据包给某个叶子节点时，路由深度(rank)为 2 的节点中有 $(M-1)$ 个可能产生误判，即深度为 2 的节点产生的额外通信开销的期望为 $(M-1)*P$ 个数据包。同理可得，深度为 i 的节点产生的额外通信开销为 $(M^{i-1}-1)*P^{i-1}$ 个数据包。因此可得整个网络由于布鲁过滤器的误报而多发生的数据包个数 S 的期望为：

$$\begin{aligned}E[S]&=\sum_{i=1}^{R-2}(M^i-1)*P^i\\&=\frac{M*P*(1-(M*P)^{R-2})}{1-M*P}-\frac{P*(1-P^{R-2})}{1-P}\end{aligned} \tag{4}$$

根据 6LoWPAN 的相关规定，一个子网中最多包含 65535 个节点 0，因此有：

$$\frac{1-M^{R-1}}{1-M}<65535 \tag{5}$$

即深度小于等于 $R-1$ 的节点之和小于 65535。

3.3.3 基于可扩展布隆过滤器的目的地址集合

布隆过滤器具有位数组大小无法动态改变的缺点。如果初始时 m 选择较小，虽然节点的存储开销相应较小，但是由式(1)可知，随着越来越多的子孙节点被插入，误报率也相应提高了，进而导致通信开销增加。反之，如果初始的时候 m 值选择过大，则会浪费节点的存储开销。本文提出了一种保障误报率上限的可扩展布隆过滤器来管理维护目的地址集合。该可扩展布隆过滤器包含多个原始的布隆过滤器，它们的误报率上限分别为 $P_0, P_0r, \cdots, P_0r^i$，其中 $0<r<1$。在插入元素时，先向第一个布隆过滤器中添加，当其误报率达到 P_0 时，创建第二个布隆过滤器，以此类推逐步增加新的布隆过滤器。检查某个元素是否存在于集合中时，需要使用每个布隆过滤器一一检测。所有的过滤器报告该元素不存在，则得出该元素不存在于集合中；否则得出该元素存在于集合中。因此该可扩展布隆过滤器的误报率为：

$$P_\xi=1-\prod_i(1-P_0r^i)\leqslant\sum_i P_0r^i \tag{6}$$

当 i 趋于无穷大的时候，P_ξ 将趋于其上界：

$$P_\xi\leqslant P_0\frac{1}{1-r} \tag{7}$$

虽然可扩展布隆过滤器可以保证误报率，但是随着过滤器数量的增加，协议的存储效率逐渐降低。为了避免可扩展布隆过滤器中的过滤器数目过多，本文提出了一种自适应的位数组大小(m)选择机制。在新一轮 DODAG 图重建时，每个节点估计出自己的子孙节点的个数，再根据该数组和误报率上限计算出 m。文献[14]得出布隆过滤器中元素的个数可以由式(8)估计：

$$n\approx-\frac{m\ln[1-\frac{X}{m}]}{k} \tag{8}$$

其中，X 为布隆过滤器的位数组中值等于 1 的个数。假设某节点的可扩展布隆过滤器中有 l 个布隆过滤器，编号从 0 到 $l-1$，对应的元素估计值为 $n_0, \cdots, n_{l-1}$，则其子孙节点数量的估计值为：

$$\hat{n}=\sum_{i=0}^{l-1}n_i \tag{9}$$

根据式(3)和式(9)可以计算出 m：

$$m=-\frac{n*\ln P_0}{(\ln 2)^2} \tag{10}$$

4 性能验证

本文搭建了一个由 51 个 Telosb[15] 节点组成的实验平台，采用 TinyOS 2.1.2[16] 操作系统，节点的部署图如图 3 所示，1 号节点为根节点。所有节点通过串口与计算机连接，并将协议开销等信息发送给计算机。由于实验场地大小有限，为了使得网络具有更多的跳数，所有节点的发射功率设置为 −10dBm，对应的发射功率等级为 11。实验过程中出现的最大跳数为 6。每个实验中，对比了非存储式和存储式 RPL，以及 B-RPL 路由在误报率 $P=0.1\%$ 和 $P=1.0\%$ 时 4 个路由协议的性能。

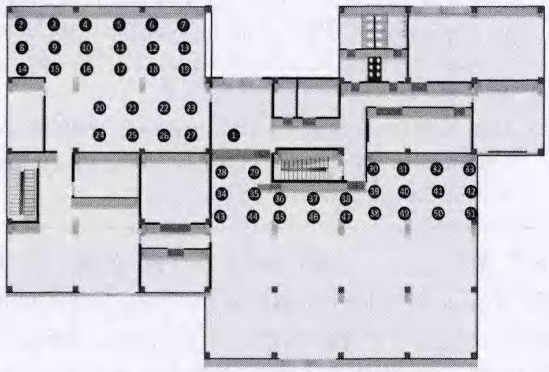

图 3 实验部署图

采用的性能指标包括通信开销和存储开销。考虑到 B-RPL 与存储 RPL 中的路由表建立过程相同，通信开销定义为将数据包从源发送到目的节点过程中所有节点在路由层发送的数据量(不考虑无线丢包和 MAC 头部)。存储开销定义为除根节点之外所有节点用于存储路由表信息的存储空间大小。

4.1 哈希函数

由于 LLNs 网络中节点(如 Telosb)的计算能力通常较差，因此本文采用了一个轻量级的哈希函数，伪代码如图 4 所示。每个哈希函数的 key 值都是随机产生的一个 32 位(bit)的无符号数。

```
1.  Function Hash( key,IPv6_addr);
2.      hash=key;i=0;
3.      for each octet in IPv6_addr;
4.          hash=hash<<i−hash;
5.          hash=hash XOR octet;
6.          i=i+1;
7.      return hash mod m;
```

图 4 B-RPL 中使用的哈希函数伪代码

4.2 P2MP 通信

待网络稳定之后，根节点发送一个有效载荷为 20 字节的数据包给每个叶子节点。图 5 展示了 4 种路由协议的通信开销。可以看出，非存储式 RPL 的通信开销远大于其它 3 个路由协议。当 P 为 0.1% 和 1% 时，B-RPL 路由的通信开销分别比存储式 RPL 多了 2.4% 和 11.8%。图 6 展示了 4 种路由协议的存储开销，非存储式 RPL 中节点不需要存储路由信息，因此存储开销为 0。此外，当 P 为 0.1% 和 1% 时，B-RPL 路由的存储开销分别是存储式 RPL 的 2.2% 和 1.6%。以 P 等于 0.1% 为例，与存储式 RPL 相比，B-RPL 以增加 2.4% 的通信开销代价换取了 97.8% 的存储空间。

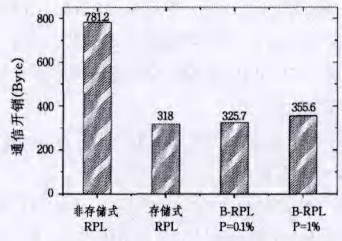

图 5 P2MP 通信下各路由的通信开销

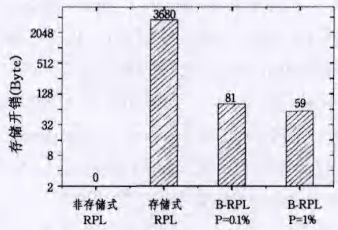

图 6 各路由协议的存储开销

4.3 P2P 通信

随机产生 100 对源和目的节点，分别运行 4 种不同的路由协议，得到的通信开销如图 7 所示。在 P2P 通信下，非存储式 RPL 和 B-RPL 的传输路径比存储式 RPL 要远，因此 B-RPL 在 P 等于 0.1% 和 1% 时的通信开销分别比存储式 RPL 多出 27.5% 和 39.5%。

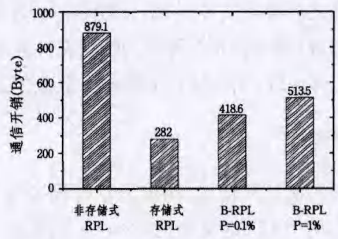

图 7 P2P 通信下各路由的通信开销

结束语 本文针对低功耗易失网络(Low-power and Lossy Networks，LLNs)中存储式 RPL 路由的大存储开销问题，提出了一种基于存储式 RPL 的低存储开销的路由协议 B-RPL。该协议充分利用了 LLNs 网络无线通信的广播特性和布隆过滤器(Bloom Filter)，极大地减少了节点的存储开销。实验显示，在 P2MP 通信模式下，B-RPL 在 P 等于 0.1% 时减少了 97.8% 的空间，而只增加了 2.4% 的通信开销。在 P2P 通信模式中，布隆过滤器的不确定性导致性能稍微有所下降。本文的下一步工作将进一步优化 P2P 通信模式下的 RPL 路由。

参 考 文 献

[1] Shelby Z, Bormann C. 6LoWPAN: The wireless embedded Internet[M]. Wiley. com, 2011

[2] Winter T, Thubert P, Brandt A, et al. RPL: IPv6 Routing Protocol for Low power and Lossy Networks [OL]. http://datatrack. ietf. ogr/dol/draftietf-roll-rpl/

[3] Kermajani H R, Gomez C. Route change latency in low-power and lossy wireless networks using rpl and 6lowpan neighbor discovery[C]// 2011 IEEE Symposium onComputers and Communications (ISCC). IEEE, 2011: 937-942

[4] Baccelli E, Philipp M, Goyal M. The P2P-RPL routing protocol for IPv6 sensor networks: Testbed experiments[C]// 2011 19th

International Conference on Software, Telecommunications and Computer Networks (SoftCOM). IEEE, 2011: 1-6
[5] 李凤国. 基于 6LoWPAN 的无线传感器网络研究与实现[D]. 南京:南京邮电大学, 2013
[6] Gaddour O, Koubaa A. RPL in a nutshell: A survey[J]. Computer Networks, 2012, 56(14): 3163-3178
[7] Gan Wei, Shi Zhi-qiang, Zhang Chen, et al. MERPL: A More Memory-efficient Storing Mode in RPL[C]// International Conference on Networks (ICON). 2013: 1-5
[8] Accettura N, Grieco L A, Boggia G, et al. Performance analysis of the RPL routing protocol[C]// 2011 IEEE International Conference on Mechatronics (ICM). IEEE, 2011: 767-772
[9] Tripathi J, de Oliveira J C, Vasseur J P. A performance evaluation study of RPL: routing protocol for low power and lossy networks[C]// 2010 44th Annual Conference on Information Sciences and Systems (CISS). IEEE, 2010: 1-6
[10] Ko J, Dawson-Haggerty S, Gnawali O, et al. Evaluating the Performance of RPL and 6LoWPAN in TinyOS[C]// Workshop on Extending the Internet to Low Power and Lossy Networks (IP SN2011). 2011
[11] Brachman A. RPL Objective Function Impact on LLNs Topology and Performance[M]// Internet of Things, Smart Spaces, and Next Generation Networking. Springer Berlin Heidelberg, 2013: 340-351
[12] Bloom B H. Space/time trade-offs in hash coding with allowable errors[J]. Communications of the ACM, 1970, 13(7): 422-426
[13] ZigBee Alliance. ZigBee specification. [OL]. 2006. http://www.zigbee.org
[14] Swamidass S J, Baldi P. Mathematical correction for fingerprint similarity measures to improve chemical retrieval[J]. Journal of chemical information and modeling, 2007, 47(3): 952-964
[15] Polastre J, Szewczyk R, Culler D. Telos: enabling ultra-low power wireless research[C]// Fourth International Symposium on Information Processing in Sensor Networks, 2005 (IPSN 2005). IEEE, 2005: 364-369
[16] Levis P, Madden S, Polastre J, et al. TinyOS: An operating system for sensor networks[M]// Ambient intelligence. Springer Berlin Heidelberg, 2005: 115-148

(上接第 91 页)

当每个线程处理完所有的 20 个 superblocks 之后(每次读取的数据是 80 个 superblocks,开启 4 个线程,每个线程处理 20 个 superblocks),写文件。即在 4 线程同步处理 80 个 superblocks 的数据之后再写文件,与串行程序的边压缩边写文件不同。经过验证(将并行程序得到的压缩文件解压缩,与原始的 FASTQ 文件比较,无差异),这种方法能进行正确的压缩。

3 结果和分析

在如下 2 种测试环境中,比较了串行程序与多线程程序压缩不同大小的 FASTQ 文件的 Process 函数运行时间。测试环境 1 为操作系统:Ubuntu 11.04,内核 2.6.38,GCC 版本号 4.5.2,8 核处理器,CPU 型号:Intel(R) Xeon(R), X5550, 2.67GHz,内存:16GB。测试环境 2 为操作系统:Ubuntu 10.10,内核 2.6.35,GCC 版本号 4.4.5,8 路 8 核,CPU 型号:Intel(R) Xeon(R) X7550, 2.00GHz,内存:512GB。

测试结果如表 1 所列。

表 1 串行、并行 DSRC 压缩时间比较

数据文件[5]	测试环境 1			测试环境 2		
	DSRC (单线程)	Pthdsrc (4 线程)	加速比	DSRC (单线程)	Pthdsrc (4 线程)	加速比
SRR013951_2. filt. fastq (3GB)	78	20	3.9	107	30	3.6
SRR765989_1. filt. fastq (7.3GB)	268	76	3.5	384	110	3.5
SRR741385_2. filt. fastq (21GB)	—	—	—	1155	321	3.6

DSRC:串行 DSRC 算法。Pthdsrc:基于 Pthreads 的并行 DSRC 算法。分别使用测试环境 1 和环境 2 的单线程和 4 线程,得到表 1 所列的结果。时间单位是秒。因为测试环境 1 的内存是 16GB,而 SRR741385_2. filt. fastq 文件是 21GB,所以没有在测试环境 1 下压缩 SRR741385_2. filt. fastq 文件。

在测试环境 1 下,压缩 SRR013951_2. filt. fastq,串行 DSRC 程序执行 Process 函数的时间是 78 秒。使用 2 个处理器核,Process 函数的执行时间是 38 秒,Process 函数的加速比是 2。使用 4 个处理器核,Process 函数的执行时间是 20 秒,Process 函数的加速比是 3.9。对于单个 superblock 的处理时间,在串行 DSRC 中是 0.08 秒,在 Pthdsrc 中是 0.02 秒。并行程序有几乎线性的加速比。

在测试环境 2 下,使用 4 线程分别压缩 3 种大小不同的 FASTQ 文件,相对于串行程序,加速比是 3.5。压缩 SRR765989_1. filt. fastq(7.3GB)文件,使用 8 个处理器核,并行算法的 Process 函数的时间是 185 秒。使用 10 个处理器核,并行算法的 Process 函数的时间是 181 秒,相比使用 4 个处理器核的情况,性能下降。

基于 Pthreads 的并行 DSRC 压缩程序,在测试环境 1 下,使用单线程、2 线程、4 线程,Process 函数有几乎线性的加速比。在测试环境 2 下,使用单线程、4 线程,Process 函数有接近3.5的加速比。当线程数增加到 8 或者 10 时,相对于 4 线程,性能下降。可能的原因是内存的限制以及通信开销的增加抵消了多线程处理带来的速度提升。

结束语 本文以串行 DSRC 算法为研究对象,基于 Pthreads 技术优化 DSRC 算法,测试结果表明,并行 DSRC 算法性能提升。下一步研究将此 Process 函数模块嵌入至进程与线程的混合并行 DSRC 算法优化中,以促进压缩算法并行效率的整体提升。

参 考 文 献

[1] Tembe W, et al. G-SQZ: compact encoding of genomic sequence and quality data[J]. Bioinformatics, 2010, 26(17): 2192-2194
[2] Jones D C, et al. Compression of next-generation sequencing reads aided by highly efficient de novo assembly[J]. Nucleic Acids Res., 2012, 40(22), e171
[3] Deorowicz S, et al. Compression of DNA sequence reads in FASTQ format[J]. Bioinformatics, 2011, 27(6): 860-862
[4] https://computing.llnl.gov/tutorials/pthreads/#Abstract
[5] http://www.1000genomes.org/