一种改进的基于延迟的TCP拥塞避免算法*)

易发胜 夏梦芹 王 焱 曾家智
(电子科技大学计算机学院 成都 610054)

摘 要 基于延迟的TCP拥塞避免算法(DCA)提高了系统的吞吐量,但在某些情况下DCA表现出较差的性能。通过对RTT的分析发现,变化的传输延迟和延迟ACK将对使用RTT指示拥塞引入明显误差,从而影响DCA算法的准确性。为此提出了一种改进的DCA算法,在判断网络是否拥塞时,先消除这些明显误差。仿真试验表明,该算法更加准确地监测到网络拥塞,改进了TCP性能,提高网络的吞吐量。

关键词 TCP,DCA,拥塞控制,拥塞避免

An Improved DCA Algorithm for TCP

YI Fa-Sheng XIA Meng-Qin WANG Yan ZENG Jia-Zhi
(Department of Computer Science, UEST of China, Chengdu 610054)

Abstract The delay-based congestion avoidance(DCA) algorithms for TCP increases system's throughput. But in some cases, the DCA presents bad performance. By analyzing RTT's component, it is found that various transmitting delay and delayed ACK bring obvious errors in using RTT as the signal of network congestion, and affect DCA algorithms' veracity. In order to solve the problem, an improved DCA algorithms is proposed, which eliminates these errors before estimating network whether or not to be congestion. The performance of the algorithm has been tested and evaluated on NS simulator. The simulation results demonstrate that the algorithm inspects network congestion more exactly. So it enhances TCP efficiency and increases good throughput of TCP flow.

Keywords TCP, DCA, Congestion control, Congestion avoidance

1 引言

因特网是基于尽力转发的IP协议来传递报文的,这样能够最大限度地利用网络资源,节省通信成本。由于缺乏有效的流量管理,因特网容易在网络的某些地方产生拥塞。TCP/IP协议栈在主机端利用TCP协议来处理这个问题。TCP是通过检测每个发送出去的数据的ACK应答来了解网络的拥塞情况的,发送方每发送一个数据报文,启动定时器,如果在超时限之前没有收到ACK,发送方认为丢包,网络发生拥塞,则减小发送窗口,降低发送速度,从而缓解拥塞。目前Internet网络主要采用Jacobson提出的Reno TCP协议[1,2],它使用滑动窗口协议,由慢启动、拥塞避免、快速重传和快速恢复4个阶段组成。

虽然TCP的拥塞控制机制有效避免了网络拥塞崩溃的发生,但是很多研究表明,TCP的拥塞控制策略趋于保守,在很多时候限制了网络的吞吐量。为了提高TCP协议的吞吐量,目前对TCP拥塞控制方面的研究主要集中在两个方面。一是出现ACK超时的时候能够快速恢复发送窗口大小,尽量提高吞吐量;二是在拥塞避免阶段监测网络的拥塞情况,在出现拥塞丢包以前,主动降低发送窗口的大小,避免网络拥塞的出现,维持TCP的高吞吐量。

前一方面的研究比较多[3~9],这些文献针对无线网络容易出错[6,7]、高速网络突发性拥塞[8]以及Web应用[9]等情况下的慢启动带来的吞吐量问题进行了深入的研究,提出了许多具有建设性的解决方案。后一方面的研究主要是基于延迟的拥塞避免(DCA)算法研究[10~12],DCA算法的代表是TCP Vegas[10]和Dual[11]。DCA算法检测数据包的RTT,对RTT的增加做出响应,在网络出现拥塞之前试图避免拥塞的发生。事实证明,DCA算法较好地改善了TCP的吞吐量。但是,在文[12]中提出在某些情况下(特别是高速链路)利用RTT来指示拥塞反而会降低TCP吞吐量。通过大量的测试分析,发现某些情况下仅有7%~18%的报文丢失和RTT的增加有关。在这种情况下,如果利用DCA算法来处理拥塞避免,将会降低TCP吞吐量达7%~58%。

本文通过对RTT的认真分析,发现影响DCA算法准确度的重要原因在于只是以RTT作为拥塞控制信号,而没有考虑到RTT中的非拥塞延迟对RTT的影响。通过分析发现,在RTT的组成中,和报文长度有关的传输延迟以及延迟ACK是RTT产生变化的重要原因,并影响了对拥塞信号的指示。基于这些分析,本文提出了一种改进的DCA的拥塞避免算法,该算法消除了影响RTT的非拥塞延迟影响,较好地反映了网络的拥塞情况,从而对网络拥塞情况有了更好的估计和反应,大大提高了TCP在拥塞避免阶段的吞吐量。仿真试验表明,该算法能够显著提高TCP的吞吐量,改善网络性能。

2 影响RTT的因素分析

在TCP协议中,RTT是发送方发送一个数据报文到收到其ACK确认的时间。通常,这个时间包含如下四个部分:数据传播时间T_p、处理时间T_d、传输时间T_t和排队时间T_q。即

$$RTT = T_p + T_t + T_d + T_q \quad (1)$$

其中传播时间T_p是数据报文在物理线路上传播所需要的时间。不同的介质传输数据的速度不一样,在电缆中,这个速度是光速的2/3,而无线介质的速度接近光速。在RTT中T_p所占的比重和传输距离有关。在同一个TCP连接过程中,

*)国家自然科学基金资助项目,编号:69871005。易发胜 讲师,博士研究生,主要研究方向:新型网络体系结构。夏梦芹 博士研究生,主要研究方向:新型网络体系结构。王 焱 博士研究生,主要研究方向:新型网络体系结构。曾家智 教授,博导,主要研究方向:计算机网络与通信。

T_p 一般都是比较固定的。虽然 IP 网络并不能保证同一个 TCP 连接中的每一个报文都走一样的物理线路，但是根据研究网络在一段时间内的状态是相对稳定的[13]，其通过的路径变化很小，T_p 几乎保持稳定。

而处理时间 T_d 是数据报文在路由器和主机上进行处理的时间总和。基本上，对于同一个 TCP 数据流，其各个数据报文，在各个路由器上的处理时间变化不大，都是同样的数据类型，实行同样的转发处理。但是在目的主机上的处理，由于 TCP 协议允许捎带延迟，有些 ACK 可能会延迟 0～200ms，这会对整个 RTT 产生较大的影响。

对于 Td，我们可以用如下公式来表示

$$Td = T_d' + T_{delay} \tag{2}$$

其中 T_d' 是各个网络设备处理数据报文的时间总和，而 T_{delay} 是捎带延迟的时间，可能在 0～200ms 之间变化。

传输时间 T_t 是数据报文中的每一位送到物理链路上的时间总和，T_t 和链路的数量 n、链路的带宽 B 以及数据报文的长度 L_p 有关。在同一个 TCP 连接中，在一定时间范围内，链路的数量和各链路的带宽保持稳定的情况下，传输时间随着数据报文的长度变化而变化。对于某一段连路的 $T_{ti} = L_p/B_i$，在有 n 个链路时，$T_{tp} = \sum_{i=1}^{n}(L_p/B_i)$。对于 TCP 的 RTT 而言，$T_t$ 还包含 ACK 返回的时间，ACK 报文一般比较短，但有时是捎带确认，其报文也可能很大。此外，ACK 报文可能走另外一条区别于数据报文的路线，设含有 ACK 的报文长度为 L_a，其经过的链路数量为 m，每段链路的带宽分别为 B'_i，则 ACK 报文的传输时间 $T_{ta} = \sum_{i=1}^{m}(L_a/B'_i)$。

因此，在 RTT 中，总的传输时间是

$$T_t = T_{tp} + T_{ta} = \sum_{i=1}^{n}(L_p/B_i) + \sum_{i=1}^{m}(L_a/B'_i) = L_p\sum_{i=1}^{n}(1/B_i) + L_a\sum_{i=1}^{m}(1/B'_i) \tag{3}$$

我们可以把 $\sum_{i=1}^{n}(1/B_i)$ 和 $\sum_{i=1}^{m}(1/B'_i)$ 分别看作带宽为 B 和 B' 的等效链路。这时，

$$T_t = L_p/B + L_a/B' \tag{4}$$

事实上，目前 Internet 的带宽在接入网比较小，而骨干网上很宽。因此，B 和 B' 接近接入网络的瓶颈带宽。这时可以假定 $B=B'$，公式(4)可以进一步简化为

$$T_t = (L_p + L_a)/B \tag{5}$$

从公式(5)可以看出，在瓶颈带宽不是很大时，数据报文和确认报文长度变化对 RTT 将产生较大影响。比如接入网络是 1Mbps 的 ADSL，在最小的报文情况下，$L_p + L_a$ 大约 1100bit，这时，传输延迟大约为 1ms；但是在最大报文的情况，$L_p + L_a$ 可达到 24000bit，传输延迟大约为 24ms，二者相差很大，由此可见，如果 RTT 相对较小的时候，传输时间的动荡将对 RTT 产生很大影响。

排队时间 T_q 是数据报文在网络设备或者主机上等待处理或者等待发送的时间。这个时间真正反映了网络的拥塞情况。网络设备或者主机只有在来不及处理报文或者来不及发送报文的时候才让数据报文等候排队。如果排队等候的报文实在太多，则产生丢失。因此如果能够比较准确地掌握 T_q 的大小，则能够比较好地掌握网络拥塞状况。

排队时间的长短在一些网络设备上和数据报文的长度有关，如果这些设备采用了 WFQ 或者 W_2FQ 等调度算法的话。另外，排队时间导致的延迟也给用户链路等效带宽降低了的感觉。我们可以采用 L/B_e 来表示一个排队时间，其中 B_e 表示和排队时间等效的传输时间。

3 基于迟延的拥塞避免改进算法

3.1 基本原理

通过前面的分析我们知道，在 TCP 的 RTT 中，能够有效体现网络拥塞情况的组成部分是排队延迟时间 T_q。但是在整个 RTT 的组成中，T_q 仅仅占有其中的一部分。在一次 TCP 连接过程中，我们并不能根据 RTT 来估算 T_q 的大小。传输延迟和延迟 ACK 会对 RTT 产生较大的变化，影响直接使用 RTT 来指示网络拥塞的精度。因此在基于延迟的拥塞避免算法中，我们必须克服这两种延迟带来的影响，提高预测拥塞的精确度。

根据前面的分析，在一定的稳定时间内，TCP 流的一个 RTT 时间可以简化如下：

$$RTT = T_p + (L_p + L_a)/B + T_d' + T_{delay} + T_q \tag{6}$$

在公式(6)中，我们需要知道 T_q 来预测网络的拥塞状况。但是要知道 T_q 是很难的，公式中我们只能够知道 RTT、L_p、L_a 等。

为了得到反映网络拥塞的量度，我们可以将相邻两次 RTT 的值作比较，即：

$$RTT_1 = T_{p1} + (L_{p1} + L_{a1})/B + T_{d1}' + T_{delay1} + T_{q1} \tag{7}$$

$$RTT_2 = T_{p2} + (L_{p2} + L_{a2})/B + T_{d2}' + T_{delay2} + T_{q2} \tag{8}$$

一般地，在相邻两次 RTT 的时间内，$T_{p1} \approx T_{p2}$，$T_{d1}' \approx T_{d2}'$。用公式(7)～(8)得：

$$RTT_2 - RTT_1 = ((L_{p2} + L_{a2}) - (L_{p1} + L_{a1}))/B + T_{delay2} - T_{delay1} + T_{q2} - T_{q1}$$

从而我们有：

$$T_{q2} - T_{q1} = RTT_2 - RTT_1 - ((L_{p2} + L_{a2}) - (L_{p1} + L_{a1}))/B - (T_{delay2} - T_{delay1}) \tag{9}$$

$L_p + L_a$ 是一次 RTT 采样时间内，发送数据报文和接收到的 ACK 数据报文的数据长度之和，用 L 表示，则公式(9)可以简化为：

$$T_{q2} - T_{q1} = RTT_2 - RTT_1 - (L_2 - L_1)/B - (T_{delay2} - T_{delay1}) \tag{10}$$

在公式(10)中，通过两次连续的 RTT 可以得到排队时间的变化情况 $T_{q2} - T_{q1}$，利用其作为网络拥塞程度的指示，将大大改善网络拥塞预测的准确性。不过，公式中 T_{delay} 无法估计，参数 B 很多情况下也不能直接得出，下面分别处理这两个问题。

对于 T_{delay}，我们只是稍微修改一下 TCP 协议，利用 TCP 首部的保留域将延迟 ACK 等待的时间值(ms)传递回来。在接受方每次发送 ACK 时，TCP 将“延迟 ACK”定时器的值填写在这里。TCP 首部保留域只有 6bit，我们定义一个单位表示 4ms，这样总共可以表示 $2^6 \times 4$ ms，即 256ms。对于一般的 TCP 算法，完全可以不理睬这个域的值，在发送方的处理是兼容的。

对于参数 B，可以利用如下办法估计：连续发送两个长度不一样的报文，可以假定此时网络的拥塞情况是比较一致的，因此

$$RTT_2 - RTT_1 = (RTT_2 - RTT_1) - (L_2 - L_1)/B - (T_{delay2} - T_{delay1}) = 0$$

从而计算出估计的 B

$$B = (L_2 - L_1)/((RTT_2 - RTT_1) - (T_{delay2} - T_{delay1})) \tag{11}$$

在实际环境中，为了防止 TCP 流的网络环境的变化，需要定期重新计算 B 的值，以保证 $T_{q2} - T_{q1}$ 的可靠性和准确性。另外，连续两个报文的长度差别越大，基于测量 RTT 而估算出的 B 误差越小。

3.2 基于迟延的拥塞避免改进算法

基于前面的分析和上述推导，我们对 TCP 的拥塞控制过程的拥塞避免阶段的算法进行了改进，称为 EDCA。定义如下：

1. 在 TCP 建立连接的过程中，根据公式(11)，利用连续的两个报文长度差别大的 RTT 计算等效带宽 B 的值。

2. TCP 在拥塞避免阶段根据公式(10)计算：

$$T_{q2}-T_{q1}=(RTT_2-RTT_1)-(L_2-L_1)/B-(T_{delay2}-T_{delay1})$$

3. 根据计算的结果，进行下面的处理：

if$(T_{q2}-T_{q1})\leqslant\alpha$ then

按照原拥塞控制算法增长拥塞窗口

else if $(T_{q2}-T_{q1})\geqslant\beta$

减小拥塞窗口 $w\Delta Q_{k+1}$/timeout

在拥塞避免阶段，我们给了一个阈值 α 和 β，如果$(T_{q2}-T_{q1})$的结果在这两个值之间，保持拥塞窗口不变，避免细小误差带来不必要的网络变化；在$(T_{q2}-T_{q1})$小于 α 时，表示网络拥塞情况减轻，可以增加发送流量。在$(T_{q2}-T_{q1})$大于 β 时，表示排队等待时间增加，网络拥塞趋势增加了，这时应减少发送方的拥塞窗口大小，从而减缓了产生拥塞趋势，让网络维持在高吞吐量状态。

至于需要减少拥塞窗口的数量，我们使用一种简单的比例算法。我们知道，TCP 在超时的时候，拥塞窗口降为零，这个时间我们称为 timeout。设当前的拥塞窗口大小为 w，在我们计算的 $\Delta Q_{k+1}>0$ 的时候，假设需要减少的窗口大小为 x，我们采用同比例减小策略，可以得到 $x=w\Delta Q_{k+1}$/timeout。

4 仿真试验及性能分析

我们使用 NS2 作为仿真工具，对改进的基于迟延的 TCP 拥塞避免算法进行了分析。NS2 是 Berkeley 大学开发的仿真平台。仿真采用的网络拓扑结构如图 1 所示。

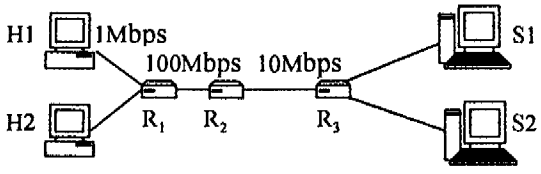


图 1 仿真试验网络拓扑

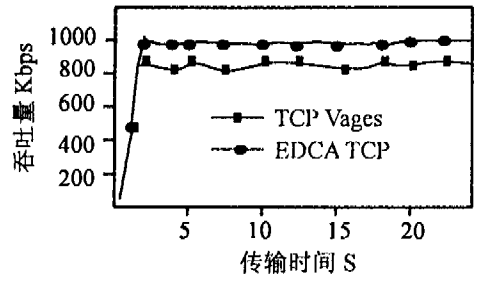
在图 1 中，共有 4 台主机和 3 台路由器。主机 H1 和 H2 通过 1Mbps 的链路和路由器 R1 相连，服务器 S1 和 S2 通过 1Mbps 的链路和路由器 R3 相连。路由器 R1 和 R2 之间通过一个带宽为 100Mbps，时延为 20ms 的链路相连。路由器 R2 和 R3 之间通过一个带宽为 10Mbps，时延为 10ms 的链路相连。路由器使用 FIFO 和 DropTail 队列管理算法，缓冲区的最大队列长度为 45 个报文段大小，接收方的通知窗口大小设为 20 个报文段。最大数据包长度分别设为 1400 字节。

下面，分别采用纯 TCP Vages 和在拥塞避免阶段采用 EDCA(下面称 EDCA TCP)来比较 EDCA 算法对拥塞避免的改进。

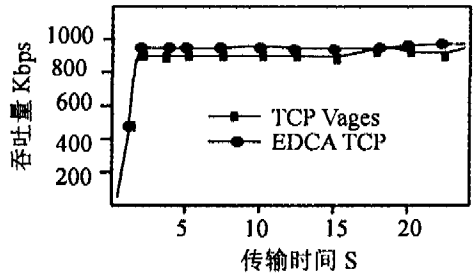
4.1 传输 Telnet 数据流

为了说明 EDCA 对传输延迟的影响具有比较好的适应性，我们构造了十个 Telnet 流，其中 5 个在 H1 到 S1 之间，另外 5 个在 H2 和 S2 之间，传输的报文在 60 至 1400 之间变化，并且所有的 TCP 流量都是双向，主机到服务器的流量是服务器到主机流量的 1/3。对于 EDCA TCP，我们设定阈值 α 和 β 分别为 -5 和 5，每发送 50 个报文重新计算等效带宽 B 的值。为了进行对比，对 EDCA TCP 和 TCP Vages 的所有其它设置都一样。

我们主要比较了 EDCA TCP 和 TCP Vages 的吞吐量情况。如图 2 所示。



(a) 传播迟延分别为 20ms 和 10ms



(b) 传播迟延分别为 80ms 和 50ms

图 2

图 2a 显示了整个网络的吞吐量情况，x 轴表示仿真时间，y 轴表示网络吞吐量变化。由图可以看出 EDCA TCP 明显提高了整个系统的吞吐量。同时我们发现，TCP Vages 吞吐量不稳定，而 EDCA TCP 相对比较稳定。这主要是由于传输延迟和延迟 ACK 的变化会给 TCP Vages 带来误判，从而影响了预测拥塞的准确性。

我们稍微调整了仿真条件，即把 R1 到 R2 的链路时延改为 80ms，R2 到 R3 的链路时延改为 50ms，这时的线路吞吐量如图 2b 所示。我们发现，TCP Vages 吞吐量和 EDCA TCP 相差减小了，这主要是由于由于数据报文长度引起的传输延迟在整个 RTT 中的比例减小了，使它们导致的结果比较相近了。如果 EDCA TCP 通过合理调节阈值 α 和 β，会改善系统的吞吐量。

4.2 传输 ftp 数据流

在实际的网络环境中，总是需要传输很多不同的文件资料。本组实验采用 ftp 传输不同大小的文件，来比较 EDCA TCP 和 TCP Vages 的性能。

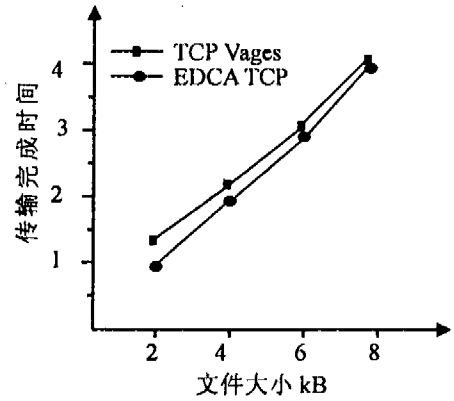


图 3 传输不同大小文件时的性能比较

我们在 H2 和 S2 中设定了 600kbps 的 UDP 背景流量，

其他环境设置不变。让 H2 和 S2 之间建立一个 TCP 流，每个 TCP 连接分别传输 2kB，4kB，6kB，8kB 大小的文件各 20 次，记录文件传输完成的时间。

图 3 是仿真试验的结果。从图中我们可以看到，文件越小，TCP Vages 所花的时间相对越多些，随着文件的增加，二者的差别越来越小。这是因为，文件越大，大多数报文都是最大报文长度，两种算法的差距就很小了。同时对 ftp 协议而言，传输文件差不多是单向数据流，也没有延迟 ACK 干扰了。

结束语 在基于迟延的拥塞避免算法中考虑传输迟延和延迟 ACK 的影响，可以提高预测网络拥塞的准确性。本文经过分析，考虑了不同报文长度以及捎带 ACK 对 RTT 采样的影响，改进了 DCA 的算法，提高了这种算法的性能。仿真试验证明，在网络数据报文长度经常变化的传输流中，本算法表现出很大的性能改进，能够明显提高网络的总吞吐量，增加 IP 网络的资源利用率。

在仿真试验中我们还发现，计算出的等效带宽 B 有时候差别较大，但并不影响本算法的最终优势。因为在网络拥塞时，计算的等效带宽明显偏小，但网络不拥塞的时候，计算出 $T_{q2}-T_{q1}<0$，根据算法，我们需要增加流量，从而保持了高吞吐量。当然，在这个时候我们需要重新估计等效带宽的大小，以便计算的拥塞迟延偏差更加准确。

参 考 文 献

1 Jacobson V, Karels M J. Congestion avoidance and control. IEEE/ACM Trans on Networking, 1988, 6(3): 314 ~ 329

2 Stevens W. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. Internet Network Working Group, RFC 2001, 1997

3 Xu W, Qureshi A G, Sarkies K W. Novel TCP congestion control scheme and its performance evaluation Communications. IEE Proceedings, Aug. 2002, 149(4): 217～222

4 Leung Fei Peng. A novel fair bandwidth allocation algorithm for TCP window control V. C. M. In: Performance, Computing, and Communications Conference, 2003. Conference Proceedings of the 2003 IEEE International, April, 2003. 317～324

5 Wu Cheng-Shong, Hsu Ming-Hsien, Chen Kim-Joan. Traffic shaping for TCP networks: TCP leaky bucket. TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. 2002, 2(28-31): 809 ～ 812

6 Chan A C F, Tsang D H K, Gupta S. TCP (transmission control protocol) over wireless links. Vehicular Technology Conference, IEEE 47th On 1997, 3: 1326～1330

7 Srivastava A, Friday R J, Ritter M W, Filippo W S. A study of TCP performance over wireless data networks. Vehicular Technology Conference, 2001. VTC 2001 Spring. IEEE VTS 53rd, 2001, 3(6-9): 2265～2269

8 李云，陈前斌，隆克平，等. 一种基于链路带宽估计的 TCP 慢启动算法. 计算机学报，2003，26(6)：693～700

9 陈晶，郑明春，孟强. 一种基于历史连接的网络拥塞控制算法及其性能分析. 计算机研究与发展，2003，40(10)：1470～1475

10 Brakmo L S, Peterson L L. TCP Vegas: end to end congestion avoidance on a global Internet. Selected Areas in Communications, IEEE Journal on, 1995, 13(8): 1465～1480

11 Wang Z, Crowcroft J. Eliminating periodic pACKet losses in the 4.3-Tahoe BSD TCP congestion control algorithm. Comput. Commun. Rev., 1992, 22(2): 9～16

12 Martin Jim, Nilsson Arne a, Rhee Injong. Delay-Based Congestion Avoidance for TCP. IEEE/ACM Transactions On Networking, 2003, 11(3): 356～369

13 paxson V. Measurements and analysis of end-to-end Internet dynamics: [Ph. D dissertation]. UC Berkeley, 1996

(上接第 37 页)

当 D 变大时，引入下一代的随机抗体数量得到增加，这主要是为了增加多样性，它可以有助算法陷入局部最优。但从图 5 看，参数 D 对算法的收敛速度并没有明显的影响，并且算法的执行时间也没有什么明显的变化。

4.3 与 DBC 算法的性能比较

为了考察 GISA 算法的性能，我们选择 DBC 算法为比较对象。二者的任务数和计算资源数均分别为 200 和 10，总预算为 6000。表 1 比较了两种算法的性能，数据为 20 次仿真实验的平均值。

表 1 2 种算法的性能比较

	DBC 算法	GISA 算法
平均任务完成时间	2727	2148
算法本身执行时间	19	186

可以看出，GISA 算法由于引入免疫原理，使得任务完成时间有了明显的提高。但算法本身的运行时间却比 DEC 算法高出一个数量级，这是算法本身的搜索特性决定的。对于大规模的科学计算问题，这点时间上的开销，却可以带来系统整体运行效率的极大改善。

结论 本文针对网格资源管理系统的任务分配调度问题进行研究，提出了 GISA 算法。其核心思想是用人工免疫系统求解任务分配问题。作为一种智能优化搜索策略，人工免疫系统(AIS)在函数优化、组合优化、调度问题等方面得到了广泛应用并取得了很好的效果。在大多数的情况下，免疫算法取得了比现有启发式算法更好的求解结果[9]。

GISA 算法考虑到了网格任务调度问题中存在的很多影响任务运行速度的因素，例如：网络线路带宽、资源的负载等等，而将任务的运行时间作为模糊时间。因此，算法就是对具有模糊运行时间的任务进行分配调度，并在用户预算约束条件下，通过免疫原理来求解该 NP 完全问题。算法从仿真实验来看，是可行和有效的，并且收敛速度较快(一般来说，比遗传算法较快)。理想中的网格应该还具有资源预定功能。因此，我们下一步的研究工作就是将资源预定功能结合到算法中，以期得到较理想的调度算法。

参 考 文 献

1 Foster I, Kesselman C. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1998

2 Buyya R, Abramson D, Giddy J. Grid Resource Management, Scheduling, and Computational Economy. International Workshop on Global and Cluster Computing, Japan, 2000

3 Baker M, Buyya R, Laforenza D, The Grid: International. Efforts in Global Computing. Intl. Conf. on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, Rome, Italy, 2000

4 Abramson D, Buyya R, Giddy J. A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker. Future Generation Computer Systems Journal, Volume Elsevier Science, 2002, 18(8): 1061～1074

5 Buyya R, Abramson D, Giddy J. An Economy Driven Resource Management Architecture for Global Computational Power Grids. Intl. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), Las Vegas, USA, 2000

6 Ibarra O H, Kim C E. Heuristic algorithms for scheduling independent tasks on non-identical processors. Journal of the ACM, 1997, 24(2): 280～289

7 De Castro L N, Von Zuben F J. Clonal selection algorithm with engineering applications. GECCO 2000 Workshop proceedings, 2000. 36～37

8 Buyya R, Abramson D. An Economy Driven Resource Management Architecture for Global Computational Power Grids. Intl. Conf. on Parallel and Distributed Processing Techniques and Applications, Las Vegas, USA, 2000

9 Buyya R, Murshed M. A Deadline and Budget Constrained Cost-Time Optimize Algorithm for Scheduling Parameter Sweep Applications on the Grid. GridSim Toolkit Release Document, Dec. 2001

10 肖人彬 王磊. 人工免疫系统：原理、模型、分析及展望. 计算机学报，2002，25(12)：1281～1293