基于R的并行统计计算

宋 磊 尹俊平 陈 虹
(北京应用物理与计算数学研究所高性能计算中心 北京 100092)

摘 要 随着统计分析中数据规模和复杂性的不断增加,高性能计算也开始在金融、经济和管理等统计计算主导的领域中发挥重要的作用。将对基于R的统计分析中并行计算技术的发展现状和最新进展做一个综述,重点从用户的角度考察R在不同体系结构计算平台上并行统计计算的实现。一个人造和真实应用的测试表明了其应用效果。

关键词 R,统计分析,高性能计算,并行统计计算
中图法分类号 TP31 **文献标识码** A

Parallel Statistical Computing with R

SONG Lei YIN Jun-ping CHEN Hong
(High Performance Computing Center, IAPCM, Beijing 100092, China)

Abstract The ever-growing size and complexity of modern data sets in statistical analysis constantly challenge the capabilities of existing statistical computing methods. High-performance statistical computing is a promising strategy to address these challenges and begins to play an important role in finance, economy, and government and so on where statistical computing is dominant. A survey on state of the art in parallel statistical computing, especially with R and within the last four years was given. Furthermore, special attention on implementations of parallel statistical computing with R under different computer systems in a users' perspective was presented. Test of synthetic code and real application show effectiveness of these implementations.

Keywords R, Statistical analysis, High performance computing, Parallel statistical computing

高性能计算在众多科学和工程计算领域中已经发挥着至关重要的作用。早期,国防安全、核爆模拟、石油勘探、天气预报等传统应用领域的需求推动了高性能计算技术的飞速发展,反过来,高性能计算又促进了这些领域应用的发展,并进一步拓展到生物信息、基因工程、医药研制、材料科学等新兴领域中[1,2]。随着统计分析中数据规模和复杂性的不断增加,这种趋势也扩展到金融、经济和管理等领域中。高性能计算技术在统计分析中的应用正日益受到相关领域研究人员和用户的关注。

本文将对基于R[3]的统计分析中并行计算技术的发展现状和最新进展做一个综述,重点从用户的角度考察R在不同体系结构计算平台上并行统计计算的实现。一个人造和真实应用的测试表明了其应用效果。

1 统计分析与高性能计算

统计分析(Statistical Analysis)是指运用统计方法及分析对象有关的知识,定量与定性相结合的研究活动,目的是从表面上看似杂乱无章的海量数据中发现和提炼出内在隐含的特征和规律。在早期,数据分析几乎等同于统计分析,随着非统计类分析方法和可视化、数据库等支撑技术的引入,目前数据分析的内涵已大为扩展。但本文除了特别说明之外,不再特别区分统计分析与数据分析。

概率论与数理统计是统计分析的数学基础。随着前者的发展和应用领域新需求的涌现,统计分析也出现了很多变化。这种变化主要体现在:1)处理对象日益复杂,数据不仅是传统地从实验、测量、观察和调查中获取的结果,还包括科学和工程计算中输出的结果;2)问题和数据的规模日益扩大,借助于传感器技术、存储技术等一系列相关技术的发展,可以解决的问题规模和需要分析的数据规模在相互促进中迅速增加,这也为后续的处理和分析带来了巨大的挑战;3)高性能的方法和算法相继出现,为应对挑战,一方面研究新的、更有效的方法,另一方面采用并行计算等技术来提升原来复杂算法的性能。高性能计算的应用是近期统计分析中最重要的趋势之一,这也反映了数据分析日益呈现出的学科交叉性。

面对一堆看似没有规律的数据,典型的统计分析通常会尝试运用多种方法并将多种手段相结合,探索性地寻找可能的特征和规律,因此这个过程是迭代的、交互的、组合运用多种分析方法的。而且,统计分析可以完成独立的分析任务,也可能作为一个复杂分析过程(例如数据挖掘,将数理统计、人工智能以及其它领域的方法用于模式发现和预测)的一个环

到稿日期:2012-10-19 返修日期:2012-12-02

宋 磊(1976-),男,硕士,副研究员,主要研究领域为科学数据管理、科学数据分析,E-mail:songlei@iapcm.ac.cn(通信作者);尹俊平(1979-),男,博士,副研究员,主要研究领域为统计数学、随机过程;陈 虹(1963-),女,博士,研究员,主要研究领域为科学数据管理、科学数据分析。

节。如果没有一个好的、支持以上特点的集成分析工具，那么这个过程将是繁琐的、效率低下的和易出错的。

因此，随着统计分析和计算机相关领域的发展，涌现出了很多面向统计分析的软件，由起初简单的算法包和专用工具到目前被统计分析人员广泛使用的综合的集成分析软件，例如 Matlab[4]、Maple[5]、SAS[6]、SPSS[7]、R[3]、S-Plus[8]，甚至 Excel 等。这些软件无论是开源的或商业的、简单或复杂的、面向统计分析或者还包含其它分析挖掘方法的，都试图为用户提供一个界面友好、编程简单、从数据输入到最后可视化整个分析过程的集成分析环境。另外，很多分析算法则是以算法库的形式提供给通晓复杂编程的分析人员，比前者更难使用。

无论是上面哪种使用方式，其核心均是统计算法的执行性能。如何将高性能计算技术应用于统计计算中？高性能计算不仅包括存储资源更多、运行速度更快、IO 和通讯性能更好的计算硬件平台（从几个核的桌面计算机、数百核的机群系统、成千上万核的大规模计算平台到网格/云计算平台），还包括匹配硬件体系结构的并行支撑软件（从并行软件库、并行编程语言到并行应用软件）。从用户角度来看，如何将原来单机运行的串行分析程序（无论是数百行的解释型语言编写的脚本或命令序列还是成千上万行高级语言编写的复杂程序）移植到多核计算平台上并具有一定的可扩展性和并行效率，这是最重要的问题。

一方面，部分统计分析程序由通晓编程的人员（多为计算机人员）采用诸如 C/C++/Fortran 等高级语言编写，并直接调用开源或商业的统计分析算法库。这类程序的并行化与其它领域程序的并行化思路和过程是一样的，本文将不做深入讨论。

另一方面，绝大部分统计分析程序（在统计分析领域中更普遍）是由具备较少编程经验的统计人员在诸如 Matlab 和 R 这类集成分析软件上采用解释语言编写的脚本程序。因为采用了更接近于自然语言的解释语言，所以这类程序与前述高级语言编写的分析程序相比，通常更为简短，但是运行性能较低，而且较紧密地依赖于集成分析软件的运行环境（部分软件提供经编译生成的可独立运行的执行代码）。对这类程序，通常采用扩展软件包的方式来提供运行环境内的并行功能，例如 Matlab 提供了 PCT 和 DCS 工具箱，R 则有超过 60 个的扩展包以不同的方式来提供并行化的支持或提高代码的性能。相比于商业软件 Matlab，开源软件 R 获得了统计分析领域更普遍的认可，尤其是在大学和研究机构。下面将介绍 R 环境上并行统计计算的实现。

2 基于 R 的并行统计计算

R 是一个包含数据处理、统计计算和图形可视化的集成统计分析系统。它最早由新西兰奥克兰大学统计系的RobertGentleman 和 Ross Ihaka 受 S 语言[9]的影响编制而成的，后来作为开源软件开放，并最终由 R 核心开发小组（R Development Core Team）继续开发和维护。它因其免费、开源、简单易用、功能强大等特点，成为统计分析以及其它相关领域内最受欢迎的软件之一，2009 年已经有大约两百万的 R 用户[10]。它不仅在大学研究机构广受青睐，而且被诸如 Google、辉瑞、美银等企业工程师用于数据分析工作。Google 首席经济学家 Hal Varian 说："R 最让人惊艳之处在于，你可以通过修改它来做所有的事情，而且你已经拥有大量可用的工具包，这无疑让你站在巨人的肩膀上工作。"，这无疑是对 R 之所以流行的最好注释[11]。R 具有如下特点：

1）R 是一门可编程（Programmable）的解释语言。R 语言正是受 S 语言和 Scheme 语言[12]的影响发展而来，语法贴近自然语言、简洁高效、易于掌握，开发效率大为提高，因此深受统计人员的欢迎。

2）可扩展，R 定义了一套可扩展包（Package）的机制，便于第三方开发和共享各种模块以扩展 R 的功能。R 的综合档案网络 CRAN[13]和 BioConductor[14]上汇集了上千由第三方开发的扩展包，内容涵盖了从统计计算到机器学习、从金融分析到生物信息、从社会网络分析到自然语言处理、从各种数据库各种语言接口到高性能计算模型。

3）R 是自由软件，不仅意味着它是完全免费、开放源代码的，还表示开发人员可以共享交流自己在 R 基础上开发的源代码。

4）R 功能丰富：强大的向量和矩阵运算功能、完整的统计分析方法、方便的数据输入输出、丰富的图形可视化，以及扩展包提供的其它众多功能。

5）R 不仅是语言，还是用该语言编写的程序可以运行的、交互的集成环境。该环境便于对新算法原型进行构建、调试、评测，也便于分析人员完成从数据输入到图形可视化分析结果的整个分析过程。

但是，R 程序的性能一直被人诟病，这几乎是所有解释型语言都存在的问题。如图 1 所示，编程语言的开发效率和执行性能是个两难问题。最近几年随着对包括研发阶段在内的整个计算程序生命周期（研发阶段和运行阶段）和开发效率的重视，高性能计算领域开始综合地看待和思考这个问题（HPC 的内涵也改为高效能计算 High Productivity Computing[15]）。

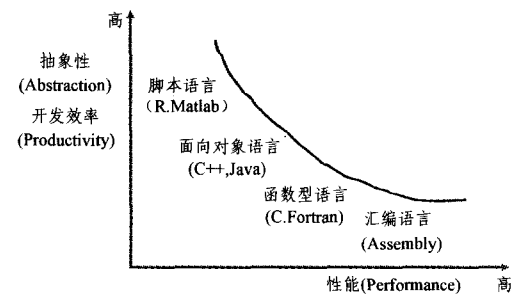


图 1 编程的两难问题

如何既保持 R 这类解释型编程带来的开发效率的提高，又能提高 R 程序的运行性能呢？在过去几年中有很多来自统计界和高性能计算界的人员致力于实现基于 R 的并行统计计算。他们的大多数成果（超过 60 个扩展包）可以在 CRAN 网站的高性能计算专题[16]上看到。文献[16]是 R 核心开发小组维护的与 R 高性能计算相关的扩展包的列表（最新版本是 2012-7-1），将其划分为显式并行、隐式并行、网格计算相关、随机数、资源管理器和调度器、应用、GPU 相关、大内存和大数据、混合语言编程以及性能分析工具等类别，对每个扩展包的介绍较为简单。2009 年的文献[17]是迄今为止对 R 程序并行较全面的综述，对主要扩展包做了较详细的介绍，

最后给出了性能比较。与该文献不同，本文将从用户的角度来考察 R 在不同体系结构计算平台上统计计算并行环境的实现，同时补充介绍近 3 年的最新进展。

用户在考虑将串行程序并行化时，面临 2 个问题：1）程序将运行在什么计算平台上；2）如何修改串行程序。问题 1）涉及不同体系结构的计算平台支持并行的硬件和软件。常用的计算平台包括多核单机系统、机群系统（Cluster）、对称多处理机（SMP）、大规模并行处理机（MPP）等多种类型[18]。但从用户使用的角度（系统镜像是否单一、内存访问是共享还是分布、通信方式是消息传递还是共享变量、并行编程接口）可以将这些计算平台分为 3 类，如表 1 所列：1）Multi-Core/Processor（多核单机系统和 SMP）：单一系统镜像，共享内存式访问，共享变量机制，Pthread 和 OpenMP；2）Multi-Node/Computer（Cluster 和 MPP）：多系统镜像，分布内存式访问，消息传递机制，MPI 和 PVM；3）Hybrid 暨 Multi-Core & Multi-Node（Cluster 和 MPP）。统计分析用户常用的是多核单机系统（桌面系统），但也逐渐拓展到小规模机群系统，不过尚未在文献中看到 MPP 上的应用。

表 1　不同体系结构计算平台比较

体系结构	Multi-Core/Processor	Multi-Node/Computer	Hybrid(Multi-Core & Multi-Node)
典型系统	多核单机系统、SMP	Cluster、MPP	
系统镜像	单一	多系统	多系统
内存访问	共享式	分布式	混合
通信方式	共享变量	消息传递	混合
编程接口	Pthread，OpenMP	MPI，PVM	混合

问题 2）暨并行方案的选择涉及 3 个方面：a）在现有的计算平台，选择哪种并行方式和扩展包；b）原串行程序的可并行特点；c）基于在选定扩展包上如何实现代码的并行，扩展包在不同的层次（例如核心算法库、用户代码）实现对并行的支持，如图 2 和表 2 所示，不仅基于的并行接口不同，而且给用户提供的接口和对程序修改的影响也不同。对于问题 a），文献[17]认为 Cluster（Multi-Node）上比较好的选择是 Rmpi 和基于 Socket 的 snow 扩展包，但因为 multicore、R/parallel 出现得较晚，而没有仔细讨论。后面的案例分析中，我们发现在 Multi-Core 类型平台上 multicore 在易用性和性能上的表现不错。multicore 是基于 POSIX 标准的 fork 调用来实现多进程并行的，派生时可完全共享父进程的状态而无需数据和代码的初始化，性能较好。在 R2.14.0 新推出的 parallel 扩展包就将 multicore 和仅支持 socket 方式的 snow 包含进去。串行程序的可并行特点（是任务并行、数据并行还是混合）也会影响选择哪种扩展包。目前大多数扩展包只支持数据并行，pR[19-22]可以支持数据并行和任务并行。对于问题 c）（见表 2），Rmpi 实现较复杂，需要用户修改大量代码，其它基本较小，pR 不需要修改。

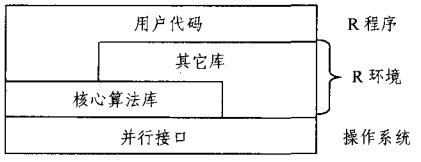


图 2　R 程序调用层次

表 2　主要并行扩展包/软件的比较

	Rmpi	Snow	pnmath	R/parallel	multicore	pR
描述	对 MPI 接口的直接包装	基于 Rmpi 的高层接口	对原数学库的并行实现	对用户代码中 loop 的自动并行化		基于即时编译（为类似中间可执行代码）的自动并行
平台	Multi-Node	Multi-Node	Multi-Core	Multi-Core	Multi-Core	Multi-Core，Multi-Node，Hybrid
基于	MPI 接口	Rmpi 等	OpenMP 接口	低层系统调用	fork	Socket，MPI
适合的并行类型	各种并行显式	数据并行	—	数据并行	数据并行	数据并行，任务并行
修改程序程度	较大很复杂	较小略复杂	很小	较小	较小	无需修改编译级自动并行

pR 是最近几年出现的、让人期待的一种 R 并行方案。它不是扩展包，而是一个基于 R 的软件框架。它是由美国能源部下属 SciDAC（先进计算中的科学发现）研究中心 DMA（数据挖掘和分析）领域研发的，目前还是非开源的状态。从仅有的几篇文献可看出，这种并行方案不同于以往的扩展包方式，而是采取对用户透明的运行时并行，暨将用户程序即时编译为某种中间代码，然后对细粒度的中间代码块进行并行处理。它的创新之处在于：1）引入并行编译技术；2）增量式源代码分析；3）对文件访问的运行时并行优化。这种方案因为不需用户修改任何代码，细粒度中间代码的并行可以支持数据并行和任务并行，所以受到了统计人员的关注。

总体来说，在 Multi-Node 类型的计算平台上比较好的选择是 Rmpi 和 snow 扩展包，前者比较复杂，适合于对并行和 MPI 接口比较精通却追求性能的用户。在 Multi-Core 平台上，multicore 在易用性和性能上的表现不错。pR 则非常让人期待。

3　案例研究

本项工作的一个驱动就是对一个有关信号预测的程序进行性能优化。程序主要有 3 部分：read 读入并处理数据；ks_check（KS 检验）判断数据块间的相似度，其核心是基于数据块某属性的聚类问题；comb_jam 核心是求 bin 的分布概率并排序，后面 2 部分是典型的统计分析程序。计算平台有 3 个，均为 Linux 系统：机器 A 是 8 核的桌面单机系统；机器 B 是 256 核的 SGI CC-NUMA 系统，共享内存、单一系统镜像，可以看作是 SMP 变种；机器 C 是数千核的某 MPP，这是最终的目标运行平台，但因时间原因本文没有给出在该系统上的测试结果。机器 A、B、C 均为 Linux 系统。

3.1　R 并行环境的构建

R 源码 2.14.2，MPI 有 MPICH2 1.4.1p1、OpenMPI 1.4.5 等，并行扩展包为 Rmpi 0.5.9、snow 0.3.8 和 multicore 0.1.7（以及 iterator，foreach，doMC，doMPI，doSNOW 等，测试结果中没有包括），其它相关扩展包有 Biobase（生物方面的数据集和算法）、prob、sm 等。

R 本身和除并行外的扩展包安装比较简单、顺利。因为并行扩展包依赖于并行接口软件包，不同系统环境可能会有不同的问题。机器 A 安装 MPICH2 较麻烦，安装后测试 Rmpi 不成功，改用 OpenMPI 后成功。机器 B 也采用 OpenMPI，顺利通过。机器 C 计算结点的系统比较精简，安装后与作业调度系统连接始终有问题，目前尚在与系统管理员寻找解决

方法。其它并行扩展包也顺利通过。

3.2 主要并行扩展包的性能比较对信号预测程序的性能优化

本节测试的程序是人工构造的，如图 3 所示，测试数据 pair 和 pair2 构造于 Biobase 扩展包。

```
>gCor<-function(x, gene=geneData){
  cor(gene[x[1],],gene[x[2],])
}
>gCor2<-function(x, gene=fData){
  md<-cbind(gene[x[1],],gene[x[2],])
  mc<-function(x,i)cor(x[i,1],x[i,2])
  boot out<-boot(md,mc,1000)
  ci<-boot. ci(boot. out,type="bca")$ bca[4 : 5]
  c(x,ci)
}
>lenght(pair)
[1] 124750
>pair2<-sample(pair,300)
>length(pair 2)
[1]300
```

图 3 人造测试程序

测试平台是机器 A，测试结果如表 3 所列。在 Multi-Core/Processor 平台上，multicore 的并行性能优于 Rmpi 和 snow，其中一个重要的原因是 multicore 共享 R 进程，因此无需拷贝数据和代码(R 函数)，后者可扩展至多结点，但均需拷贝数据和代码。此外，snow 支持(混合 windows、Linux 和 Max 系统的)异构 cluster。Revolution 公司的 foreach 及 iterator、doMPI、doMC 和 doSNOW 则是在 Rmpi、snow、multicore 基础上为进一步减少修改代码的繁琐开发的，性能会略有损失，其测试数据略去。

表 3 测例 1 在不同并行扩展包下的性能比较

(单位：秒)	gCor(pair)	gCor2(pair2)
串行	10.951	67.183
multicore(8 核)	2.173	8.976
Rmpi	2.381	9.888
snow(Socket,8 核)		10.711
snow(Rmpi,8 核)		9.923

3.3 对信号预测程序的性能优化

测例 2 即是前面提到的信号预测程序。用 R 自身的性能测试工具给出的测试结果如图 4 所示，热点在 comb_jam 和 ks_check 两个子模块上。

Total run time:7.32seconds

% total	total seconds	% self	self seconds	name
62.0	4.54	0.0	0.00	"comb_jam"
51.6	3.78	0.0	0.00	"prob"
51.6	3.78	0.0	0.00	"prob. default"
21.0	1.54	1.4	0.10	"read. table"
21.0	1.54	0.0	0.00	"read_data"
21.0	1.54	0.0	0.00	"read_filedata"
16.9	1.24	0.0	0.00	"ks_check"
16.9	1.24	0.0	0.00	"ks_check_all"
16.7	1.22	1.4	0.10	"ks_test"

图 4 信号预测程序的热点

对其优化，主要包括向量化、数组预分配、混合编程(即将计算密集程序改为 C 程序)和并行化。本测试主要关注并行化的结果。聚类算法并不是天然数据并行的，因此要仔细了解算法的领域内涵。ks_check 根据数据块之间的 KS 检验结果来判断其相似度，并根据某阈值来进行聚类。将原算法的并行可以看作是一维的数据划分，边界处需要影像数据，因此是有少量通信的数据并行。通过在起始阶段直接读入影像数据可实现完全的数据并行。comb_jam 原算法是对全场数据遍历处理的，并行比较有难度。但实际上 ks_check 正是判断数据块之间的相似度，比较相似才继续运行 comb_jam。因此，实际上可以基于数据块来并行分布计算。测试在机器 A 和 B 上分别测试，采用 multicore 并行扩展包。测试结果如图 5 所过示。

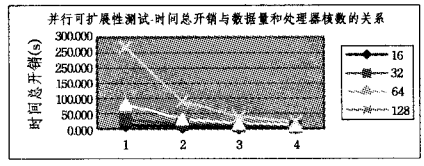
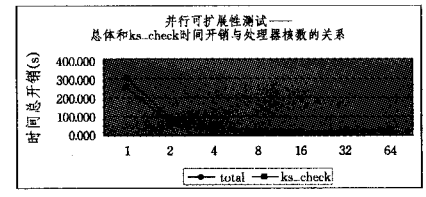
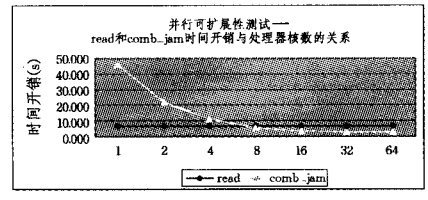


图 5 在机器 A 上分别采用不同处理器核数和不同数据量下的时间总开销比较

图 5 给出的是在机器 A 上分别采用不同处理器(1、2、4、8)和不同数据量(16、32、64、128 块)下的时间总开销，可以看出并行程序在使用最多 8 个处理器核时具有较好的可扩展性。图 6 是在机器 B 上分别使用了 1、2、4、8、16、32、64 个核的测试结果——总体和各个子模块的时间开销，其中(a)显示程序总体可以扩展到最多 16 个核。对每个子程序分别比较，read 因为没有优化，所以时间开销是大体固定的，但其在总时间所占的比例随核数增加而增大。comb_jam 的可扩展性比较好。ks_check 在 1～16 之间扩展性比较好，但随着核数的增加，每核上的数据块数急剧降低至 1，导致其时间开销在 32 和 64 反而上升，说明数据划分的粒度不能太小。由于它在总体时间开销所占的比例较大，导致整个程序的可扩展性在 32 和 64 核时不好。



(a) 总体和 ks_check 部分



(b) read 和 comb_jam

图 6 在机器 B 上分别采用不同处理器核数的开销比较

结束语 本文对基于 R 的统计分析中并行计算技术的发展现状和最新进展做了一个综述，重点从用户的角度考察 R 在 Multi-Core/Processor、Multi-Node 和 Hybrid 等计算平台上并行统计计算的实现，并采用人造和真实应用的程序做了测试。从测试可看出：1)现在 R 主要的并行方案仍主要集中在挖掘程序的窘迫并行上，但其它很多应用的通信并不是如此简单；2)现在主流的 MPP(Hybrid)计算平台上的 R 并行环境部署(与作业调度系统)仍比较麻烦。下一步的工作将关

注这两个方面。此外，pR 的实现思路比较好，值得进一步关注。

参 考 文 献

[1] 赵毅，朱鹏，迟学斌，等. 浅析高性能计算应用的需求与发展[J]. 计算机研究与发展，2007，44(10)：1640-1646

[2] 迟学斌，赵毅. 高性能计算技术及其应用[J]. 中国科学院院刊，2007，22(4)：306-313

[3] Development R，Team C R. A language and environment for statistical computing[R]. R Foundation for Statistical Computing，Vienna，Austria，2012

[4] http://www.mathworks.com/products/matlab

[5] http://www.maplesoft.com

[6] http://www.sas.com

[7] http://www.ibm.com/software/analytics/spss/

[8] http://spotfire.tibco.com/Products/S-Plus-Overview.aspx

[9] http://stat.bell-labs.com/S/

[10] Vance A. R You Ready for R? [OL]. http://bits.blogs.nytimes.com/2009/01/08/r-you-ready-for-r)

[11] Vance A. Data Analysts Captivated By R's Power[OL]. http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html

[12] http://www.schemers.org

[13] R CRAN[Z]. http://cran.r-project.org/

[14] http://www.Bioconductor.org

[15] Kepner J. HPC Productivity：An Overarching View[J]. International Journal of High Performance Computing Applications：Special Issue on HPC Productivity，2004，18(4)：393-397

[16] Cran R. Task View：High Performance and Parallel Computing with R[OL]. http://cran.r-project.org/web/views/HighPerformanceComputing.html

[17] Schmidberger M，et al. State of the Art in Parallel Computing with R[J]. Journal of Statistical Software，2009，31(1)：1-27

[18] 黄铠，徐志伟. 可扩展并行计算技术、结构与编程[M]. 北京：机械工业出版社，2000

[19] Breimyer P，et al. pR：Lightweight，Easy-to-Use Middleware to Plugin Parallel Analytical Computing with R[C]// IKE 2009. 2009：667-673

[20] Breimyer P，et al. pR：Automatic parallelization of data-parallel statistical computing codes for R in hybrid multi-node and multi-core environments[C]// IADIS AC(2). 2009：28-32

[21] Shah N，et al. pR：Enabling Automatic Parallelization of Data-Parallel Tasks and Interfacing Parallel Computing Libraries in R with Application to Fusion Reaction Simulations[C]// The !R User Conference 2010. Gaithersburg，Maryland，USA，2010

[22] Li Jiang-tian，et al. Transparent runtime parallelization of the R scripting language[J]. Journal of Parallel and Distributed Computing(JPDC)，2011，71(2)：157-168

(上接第 76 页)

由分布式字符匹配算法的硬件结构可知：与传统的串行匹配算法相比较，本文提出的分布式匹配算法可以实现并行处理，能够大大地降低算法的时间复杂度，使算法的匹配效率更高。

4 实验及分析

实验采用的硬件平台为 Xilinx ISE 13.2，基于 Xilinx 系列 Virtex5 型号的 FPGA 器件进行模拟仿真。

网络数据包的长度通常在 40～1500 字节之间，因此取不同长度的数据包进行验证。对于不同数据包长度下执行时间的测试，实验采用了 snort 系统 2012 年 3 月注册版的规则集[11]，从中提取 100 条正则表达式，每条正则表达式随机取 50 组数据包进行验证，得到在本文算法下执行时间的平均值，并与传统的算法进行比较，见表 5，表 5 中数据包长度即为 p 值。由表 5 可知，数据包长度越长，本算法的执行效率越高。

表 5 不同数据包长度执行时间

数据包长度（字节）	传统算法执行时间（时钟周期）	本文算法执行时间（时钟周期）	效率提高倍数
40	40	8	5.00
100	100	14	7.14
150	150	17	8.82
300	300	13	13.04
500	500	27	18.53
1000	1000	30	33.33
1500	1500	37	40.54

结束语 本文提出了一种基于分布式存储的正则表达式匹配算法，它能够有效地解决匹配过程中的时间复杂度问题，提高了匹配效率。实验表明，相比传统的匹配算法，该算法在处理时间上至少可以提高 5 倍以上，可以有效地实现在高速、大容量的 Internet 上对网络入侵的实时检测。目前国内外提出的算法都是针对如何压缩 DFA 空间的，尚未提出关于降低时间复杂度的算法，本文提出的算法可以实现并行处理，有效地降低了时间复杂度。下一步工作将考虑采用多核结构继续优化该算法，以进一步提高算法性能。

参 考 文 献

[1] 锐捷网络. DPI 技术白皮书[OL]. http://wenku.baidu.com/view/aa73eac66137ee06ef f91879.html，2009-03

[2] 邓凯元，姜磊. 正则表达式匹配引擎性能分析[J]. 计算机与现代化，2011，30(07)：105-107

[3] 张洁坤. 时空高效的正则表达式匹配算法研究[D]. 长沙：湖南大学，2010

[4] 刘俊超，赵国鸿，陈曙晖. 一种用于深度报文检测的 DFA 状态表压缩方法[J]. 计算机工程与应用，2008，44(22)：74-76

[5] 杨毅夫，刘燕兵，刘萍，等. 正则表达式的 DFA 压缩算法[J]. 通信学报，2009，30(10A)：36-42

[6] 姚远，刘鹏，单征，等. 面向存储的正则表达式匹配算法综述[J]. 计算机应用，2009，29(12)：3171-3173

[7] 刘胤. 深度包检测技术的研究与设计[D]. 贵阳：贵州大学，2008

[8] Sidhu R，Prasanna V K. Fast Regular Expression Matching using FPGAs[C]// The 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines. USA：Rohnert Park，California，2001：227-238

[9] Song T，Zhang W，Wang D S，et al. A memory efficient multiple pattern matching architecture for network[C]// Proceedings of the IEEE INFOCOM 2008. USA：Phoenix，2008：166-170

[10] Domenico F，Stefano G，GregorioP，et al. An improved DFA for fast regular expression matching[J]. ACM SIGCOMM Computer Communication Review，2008，38(5)：29-40

[11] Introduction to Snort [OL]. http://www.snort.org/docs/，2012-03-21