一种基于MapReduce的防火墙策略冲突并行化检测及消解模型

肖 淇 秦云川 阳王东 李肯立
(湖南大学信息科学与工程学院 长沙 410082)

摘 要 防火墙在网络安全中起到很重要的作用,其中防火墙策略中的规则决定了网络数据包被"允许"或被"拒绝"进出网络。对于大型网络来说,由于规则太多,管理者很难保证其中不出现冲突,因此策略中规则冲突的检测及解决成为了保证网络安全的重要方面。提出了一种基于MapReduce模型的防火墙策略冲突检测解决算法,它对由基于规则的分段技术得到的片段进行自定义的排序,之后将其转化为规则的形式来代替原来的规则进行数据包的过滤。片段间两两不相交且匹配的包只执行一种动作,从而消除了冲突。

关键词 防火墙,规则冲突,分段,动作,排序
中图法分类号 TP39 **文献标识码** A

MapReduce-based Parallelization Model for Firewall Policy Conflict Detecting and Resolving

XIAO Qi QIN Yun-chuan YANG Wang-dong LI Ken-li
(School of Information Science and Engineering, Hunan University, Changsha 410082, China)

Abstract Firewalls are playing a very important role in network security, because the firewall policy rules are determining that the network packet "Allow" or "Rejected" out of network. For the large networks, the rules are too many to ensure they have not conflict, therefore the detection and resolution of the policy conflict become an important aspect of network security. This paper presented a parallel method of firewall policy conflict detection and resolution algorithm, which resorts the segments formed by the rule-based segmentation technology, and translates the segments into the form of rules, uses this new rules instead of the original rules for packet filtering. Because all segments are pairwise disjointed and every segment has one action, the conflicts in policy are resolved.

Keywords Rules, Confliction, Segment, Action, Ordering

1 引言

在网络连接领域里,网络的安全问题在研究和产业界获得越来越多的关注。随着网络攻击威胁的增大,防火墙在网络中的应用成为了一个很重要的元素。防火墙就像是一个网络防护者,对于抵御攻击以及阻止未得到授权的通信往往起到很大的作用,保障了网络的安全。其中决定网络数据包是否通过防火墙的是一组有序的防火墙过滤规则。

在防火墙策略中,大量的规则之间也许会存在一些异常,即当一个网络数据包到来的时候,可能会与两条或多条规则相匹配,当匹配的规则同时存在"允许"和"拒绝"的动作时,就出现了冲突。为了保证网络的安全,对于这些防火墙规则的管理变得很重要。在一个大型的网络中,可能会存在成千上万条甚至更多的策略规则,它们可能由不同的管理者在不同的时间完成编写[1]。大量的规则导致冲突的可能性提高,使得网络安全面临巨大的威胁。

目前对于防火墙规则冲突的研究,众多研究者都已经取得了很大的成就。研究者们提出了很多针对防火墙分析的工具,比较突出的有E. Al-Shaer等人提出的Firewall Policy Advisor[1]和L. Yuan等人提出的FIREMAN[2],它们都可以用于策略异常的检测分析。Firewall Policy Advisor只能够检测成对的规则之间是否存在异常;而FIREMAN工具可以通过分析一条规则与由它之间所有的规则组成的包空间之间的关系来检测多条规则之间的异常。但是FIREMAN工具在检测异常方面也存在一些局限性[3]。每一个由FIREMAN工具得到的分析结果只能够表示某一条规则和它之前规则之间出现了异常,而不能准确地判断出现异常的规则。

防火墙策略中的冲突检测出来后,最终的目标就是解决这些冲突从而保证网络的安全。目前为止,防火墙策略的检测及解决方法的研究发展势头很迅猛。A. Hari等人提出通过增加过滤器的方式来解决规则间的冲突[4];Fu等人定义了高级别的安全要求,开发了一套机制来检测和解决IPSec规

到稿日期:2012-10-19 返修日期:2012-12-27 本文受国家自然科学基金重点项目(61133005),国家自然基金项目(61070057, 61103047),国家科技支撑计划项目(2012BAH09B02),教育部科技创新工程重大项目培育资金项目(708066),教育部博士点基金(20100161110019),湖南省杰出青年基金(12JJ1011)资助。

肖 淇(1988-),女,硕士生,主要研究方向为高性能计算,E-mail:xiaoqi.0909@163.com;秦云川(1983-),男,博士生,主要研究方向为嵌入式系统安全;阳王东(1974-),男,副教授,主要研究方向为高性能计算等;李肯立(1971-),男,教授,博士生导师,主要研究方向为并行分布式处理、高性能计算等。

则间的冲突[5]；而 Golnabi 等人则采用数据挖掘技术来解决规则冲突[6]。

除此之外，Hongxin Hu 等人研究出一种基于可视化界面的防火墙策略分析工具 FAME(Firewall Anomaly Management Environment)[7]，这个工具的框架以基于规则的分块策略为基础。冲突解决方法基于对受保护网络的风险评估和策略定义的意图分析，从而使约 92%左右的冲突得到解决。但是该工具中使用了一种重排序算法，其时间复杂度为 $O(n^2)$，当策略中的规则数量很大时，整个解决过程将变得很费时。本文提出了一种新的解决方法，即对由基于规则的分块技术得到的片段进行简单的自定义的排序来取代规则的重排序，将排序后的片段通过基于规则的分块技术的逆函数转化成为规则的表达形式，取代原来的规则。由于这些分段技术处理得到的片段是两两不相交的并且每一个片段都只有一个动作，使得冲突不再存在。本文的基本思想是在不同片段分类里利用并行化的方法来对同类型的片段进行排序，这种方法简化了 FAME 框架里的重排序算法，同时保证了防火墙策略冲突的解决。当网络数据包到来时，本算法排序后的防火墙策略能够快速、有效地决定“允许”或“拒绝”操作来保证网络安全。

2 防火墙策略异常概述

一个防火墙策略规则由一组条件来定义，当一个网络数据包符合所有定义的条件时，一个规定的动作将被执行。这组条件主要包括以下 5 个元素[8]：网络协议类型、源 IP 地址、源端口号、目的地 IP 地址和目的地端口号。这些元素作为条件来确定是否允许数据包通过[9]。表 1 是一个防火墙策略的例子，列出了策略中的 5 条规则，即 r_1, r_2, r_3, r_4, r_5。每条规则的各个条件都可以定义成一个范围内的值。如 r_5 中的协议版本号为“*”，表示协议可以为 TCP 或者 UDP，源 IP 地址 10.1.1.* 表示范围在 10.1.1.0 到 10.1.1.255 的 IP 地址。

表 1 防火墙策略示例

规则编号	协议版本类型	源 IP 地址	源端口号	目的 IP 地址	目的端口号	动作
r_1	UDP	10.1.2.*	*	172.32.1.*	53	deny
r_2	UDP	10.1.*.*	*	172.32.1.*	53	deny
r_3	TCP	10.1.*.*	*	192.168.*.*	25	allow
r_4	TCP	10.1.1.*	*	192.168.1.*	25	deny
r_5	*	10.1.1.*	*	*	*	allow

通常情况下，防火墙策略规则之间的异常包括 4 类[10]：遮蔽(Shadowing)、泛化(Generalization)、相关性(Correlation)和冗余(Redundancy)。一个规则被遮蔽指的是前面的规则能够匹配所有这个规则匹配的数据包，使得这个规则不会被激活。一个规则被泛化指的是第一个规则能够匹配所有第二个规则匹配的数据包，但是它们定义的过滤动作不相同。两个规则相关当且仅当它们有不同的过滤动作，而第一个规则匹配的一些数据包也能够跟第二个规则匹配，第二个规则匹配的一些数据包同样能跟第一个数据包匹配。一个规则冗余指的是这个规则匹配的所有数据包也能够被其它规则匹配，删除这个规则对防火墙策略没有任何影响。

以上的策略异常定义主要是针对两个规则之间的关系，对于一个防火墙策略来说，冲突的定义应该是全局的[7]。本文中明确指出了由一组有重叠的防火墙策略规则引起的冲突，这样的定义是很关键并且有助于解决冲突的。

定义 1(策略冲突[7]) 防火墙 F 中的一个策略冲突 pc，是与一组特定的有冲突的防火墙规则 $c_r=\{r_1,\cdots,r_n\}$ 相关联的，这组规则能够得到一个共同的网络数据包空间。这个空间里的所有数据包能够匹配一组相同的防火墙规则，但是至少有两条规则有不同的动作：允许和拒绝。

3 防火墙策略冲突检测及消解算法

本文提到的防火墙策略异常检测和解决算法分为 4 步：首先将所有的规则转化为数据包空间结构并进行分段，得到两两不相交的片段，片段中如果包含多个规则且规则执行的动作不一致，则表示有冲突；然后为每一个片段设置动作约束条件，使得每一个片段都有确定的一个动作，从而解决冲突；接着将这些片段采用并行化的算法进行排序，得到有序的片段序列；最后将这些片段序列转化为规则表达形式，以便于数据包的匹配。

3.1 基于规则的分段技术

为了能够解决由一组有重叠的规则引起的冲突，Hongxin Hu 等人[7]提出了一种基于规则的分段技术，即采用二叉决策图(BDD)来表示规则并进行运算，将一组规则转化为一组不相交的数据包空间。这个技术已经在几个研究领域中使用，例如网络流量测量、防火墙测试和优化。将规则转化为数据包空间的伪代码如算法 1 所示。对于一组输入的规则，将其每一条规则转化为一个数据包空间，而这个数据空间与所有已存在的数据空间之间存在 3 种关系：子集、超集和不相关，将前两种关系的重叠数据包空间进行分块合并等操作，输出一组不相关的数据包空间。算法输出的一组数据包空间片段，它们是两两不相交的，并且同一个片段里的任何两个不同的数据包匹配同一组策略。

算法 1 一组规则 R 的网络数据包空间形成算法

输入：一组规则 R

输出：一组数据包空间片段 S

```
1. foreach r∈R do
2.   s_r←PacketSpace(r);
3.   foreach s∈S do
4.     /* s_r 是 s 的子集 */
5.     if s_r⊂s then
6.       S.Append(s−s_r);
7.       s ← s_r;
8.       break;
9.     /* s_r 是 s 的超集 */
10.    else if s_r⊃_s then
11.          s_r← s_r−s;
12.    /* s_r 与 s 相交 */
13.    else if s_r∩ s ≠ Φ then
14.          S.Append(s−s_r);
```

15. $s \leftarrow s_r \cap s$;

16. $s_r \leftarrow s_r - s$;

17. S.Append(s_r);

18. return S;

3.2 策略片段的动作约束条件

在算法1转化得到的数据包空间片段中，如果有一个动作约束条件来确定该片段中数据包的动作，冲突就不存在了。对于无重叠的片段来说，片段里数据包的动作由该片段包含的规则决定；对于有重叠但无冲突的片段来说，片段里数据包的动作由该片段中包含的所有规则的共同动作来决定；对于有冲突的片段来说，它们涉及两条或多条规则且动作有冲突，因此应该为这样的冲突片段设置统一的片段约束条件。

每一个冲突片段表示一个策略冲突，里面包含的规则就是冲突里涉及的规则。Hong Hu 等人[7]提出，为每一个冲突的片段确定一个动作约束条件，使得冲突片段中涉及的数据包都统一采用该动作来解决策略冲突。一个冲突片段的动作约束条件指的是当这个片段里匹配的数据包到来时，防火墙策略应该执行的动作(允许或者拒绝)。

冲突约束条件的形成主要是基于策略的解决方法，即最小限度地与系统管理者进行交互的有效解决策略，分为手动策略和自动策略。方法对每一个冲突片段进行风险检测，得到一个风险评估值。这个值用于判断系统期望匹配这个片段的数据包执行的动作，如果片段的风险值大于给定的风险值上限(UT)，则这个片段期望的动作应为拒绝；如果片段的风险自小于给定的风险值下限(LT)，则这个片段期望的动作应为允许；如果片段的风险值在上限和下限之间，管理者应该根据这个冲突的特性手动确定片段期望的动作。其中的上限值和下限值由管理者确定。

风险值由基于受保护网络的弱点评估来决定，采用通用缺陷评估系统(CVSS)[11]来作为风险评估的安全度量。利用两个主要因素——缺陷的可利用性和缺陷的安全性，来评估网络的风险值[12]。另一个主要的因素是资产重要性值，例如系统管理者会将防护重要服务器的优先级设置得高于普通PC机，用这个值来表示对于一个网络攻击者或者管理者来说的服务器的固有值。结合CVSS的漏洞基准值(base score)(包括了缺陷的可利用性和缺陷的安全性)和资产重要性值来计算每一个漏洞的风险值，如式(1)所示：

$$RiskValue=(CVSSBaseScore)\times(ImportanceValue) \quad (1)$$

为了计算每一个冲突片段的风险值。需要积累这个冲突片段所有覆盖的漏洞的风险值，对于管理者来说，其更关心的是网络中每一个漏洞的安全值。因此，以式(2)来计算片段的风险值的平均值：

$$RL(cs)=\frac{\sum_{v\in V(cs)}(CVSS(v)\times IV(s))}{\alpha\times|V(cs)|} \quad (2)$$

式中，$V(cs)$表示 cs 冲突片段所包含的所有漏洞；$CVSS(v)$表示CVSS漏洞 v 的基准值；$IV(s)$表示服务器 s 的重要性值；变量 $\alpha(1/|V(cs)|\leqslant\alpha\leqslant 1)$允许管理者选择平均或总体风险值来评估每一个冲突片段的风险，当 α 减少时，表示管理者更注重总体风险值，当 α 增加时，说明管理者更注重平均风险值。

3.3 基于 MapReduce 的片段排序算法

对于每一个数据空间的片段，其匹配的数据包到来时都有一个确定的动作执行。由于这些片段之间是两两不相交的，对于每一个到来的数据包，或者都不匹配，或者只与一个数据包空间片段匹配，这解决了策略里的冲突。但是由于片段的数量会大于原来规则的数量，使得包的匹配更复杂。特别是规则数量很大时，片段数量过大会严重影响防火墙的数据包过滤功能。为了不影响防火墙的过滤功能，使得本文提出的方法能够适用于实际情况，本算法基于 MapReduce 模型[13,14]，采用集群计算机将得到的所有数据包空间片段进行排序，得到一个从小到大的片段顺序。

3.3.1 MapReduce 模型

MapReduce 由 Google 公司发明、主要用于处理大数据量的分布式计算模型。模型隐藏了并行化、容错、数据分布、负载均衡等细节，并将其放到一个库里，用户只需要关心他们需要执行的简单运算就行了。其主要原理如下：对于输入数据，应用 map 操作来计算出一个中间 key/value 对集，对所有具有相同 key 值的 value 值进行 reduce 操作，用户只需指定相应的 Map 函数和 Reduce 函数即可。

在大型网络中，由于规则基数大，根据规则之间相关度大小的不同，片段的数目会是规则数目的2倍甚至10倍以上，严重影响了数据包到来的匹配，而且排序算法的时间会很长。因此，本文采用的 MapReduce 模型是为了防止大数据量的计算引起的计算时间过长甚至无法处理的情况。

3.3.2 片段排序算法

在前面的3.1节提到，所有的规则都表示为 BDD 数据包空间格式的形式。每一个规则都有5个条件，表示为〈协议版本，源 IP，源端口号，目的 IP，目的端口号〉五元组。BDD 是一种用来表示布尔函数的结构，3.1节中的 BDD 结构形式的规则是将所有规则条件的布尔表达式连接在一起，形成由 0/1/X 表示的序列。这个序列中的每一位都被分配了一个布尔变量，每一个规则的布尔表达式或包含这个布尔变量或者包含它的补集。为了便于之后的排序，本文把这些布尔表达式表示为二进制数的形式，对于既可以为0也可以为1的布尔变量用“*”来表示。例如，对于表1中的 $rule_1$，“UDP 10.1.2. ** 172321.*53”，可以表示为“10000101000000001000000010 ********** 10101100001000000000001 ******** 00110101”的布尔表达式。其中，第一个 bit 值“1”表示“UDP”，本文规定，第一个 bit 值“0”表示“TCP”，“*”表示“TCP”和“UDP”的任意值。由于片段的形成是由规则布尔表达式本身或者它的差集形成，因此片段的布尔表达式或者与规则类似，或者是这些布尔表达式的合集。本算法为了简化排序算法，对于布尔表达式的合集进行简单分析且只取合集的第一个值进行排序，例如简单的 BDD 结构 $f(x_1,x_2,x_3)=x_1x_2x_3'+x_1'x_2x_3$，二进制表示方法为 110+011，由于第一位既可以为0也可以为1，将它归于“*”类别，并且取110进行排序。

在以上排序算法中，当规则数量很大时，片段的数量会非常大，可以采用基于 MapReduce 模型的并行算法来实现片段排序。图1是算法的流程图，对于输入的规则布尔表达式集合，首先在 map 阶段，可以按照片段布尔表达式的第一个字符值将所有的片段分为3大类，即“TCP”类型片段(第一个字

符为“0”)、“UDP”类型片段(第一个字符为“1”)和第一个字符为“*”类型的片段,输出〈片段类型,片段布尔表达式〉键值对;之后,在 reduce 阶段,将所有片段类型相同的布尔表达式进行排序,本文采用快速排序方法来进行大小排序,其中两个布尔表达式的大小这样确定:从第二个字符值开始(因为每一组内部第一个字符值相同),按照顺序 0<1<* 进行排序,如果值相等,则比较第三个字符的值,以此类推;最后在 reduce 输出阶段,将 3 类排序后的片段按第一个字符值的 0<1<* 顺序综合,得到一个按序的片段集合。

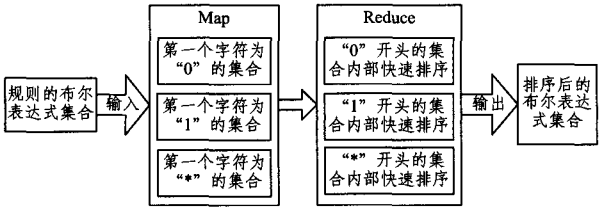

图 1 排序算法流程图

3.4 策略片段的规则形式转换

排序后的片段组都是二进制表示方法的字符串,不适用于数据包的匹配。为了对数据包进行正常的匹配,本算法将片段转化为原来的规则表示形式,即将 97bit 的布尔表达式 bit 块转换为 5 个规则的条件的形式。转化方式是规则转化为数据包空间的逆运算。同时,将该片段的动作作为该规则的动作。转换之后,由于 0<1<*(表示 0 或者 1 之中的一个)的顺序,采用第一个值匹配的策略时,特殊值(0 或者 1)排在了任意值(0 或 1 之中任意一个)前面,保证了匹配的快速和准确性。也就是说,数据包空间已经按照 TCP、UDP、任意协议类型的顺序排列,对于所有到来的 TCP 数据包,会先匹配前面的 TCP 片段而不是任意协议类型片段,后面的值的匹配也类似。这样的转换以后,由于片段之间是两两不相交的,转换得到的规则也是两两不相交的,且只有一个动作需要执行,因此解决了策略冲突。

4 仿真实验及评估

本文采用 java 语言,基于 MapReduce 模型实现该算法。仿真实验是在 8 台 Intel 8 核搭建的集群上进行。算法实现包括以下几个步骤:首先对规则进行分块运算并进行简单的处理。然后将分块后的片段加入到 MapReduce 模型中运算,map 阶段包括将输入数据按照第一个字节进行分类,共分为 3 大类;reduce 阶段分别对 3 类数据进行快速排序,并将排序后的结果转化为规则表示形式,所有数据综合在输出文件里输出。

规则的分块处理与 Hongxin Hu 等人提出的 FAME[7] 模型类似。将得到的片段转化为二进制字符串形式后,作为输入加入到 MapReduce 模型中进行排序。MapReduce 模型用于大规模数据集的并行运算,因此本算法主要用于进行大规模的规则的冲突检测和解决。在实现过程中发现,在规则数量较少的情况下,消耗的时间很长,但是对于规则数量很大时,由于处理时间增加并不多,相对于其他算法来说,在时间上有很大的优越性。图 2 是片段数目从 60 到 10040 之间排序算法所消耗的时间。由图 2 可知,当片段数目很大时,排序算法所需要的时间仍然在 16s 以内,远小于 FAME 框架中对于 926 条规则(约 2000 个片段)所需的 253.76s[7]。

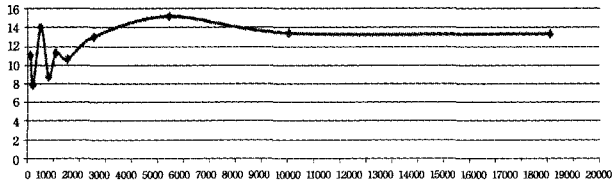

图 2 22000 片段内的算法执行时间

图 3 显示的是两种算法在规则规模相同时所需要的排序时间的比较,其中横坐标为规则数目,纵坐标为所需时间。由图中可知,当规则数目小于 150 时,FAME 框架中的重排序算法所需的时间比较少;当规则数目在 150 左右时,两种算法所需时间相似;但是当规则数目由 150 不断增大时,本文提出的基于 MapReduce 的排序算法所需的时间并没有太大的变化,而重排序算法所需的时间大大增加,特别是,本文提出的算法能够处理数以万计的规则数量规模,且算法时间相对较短。

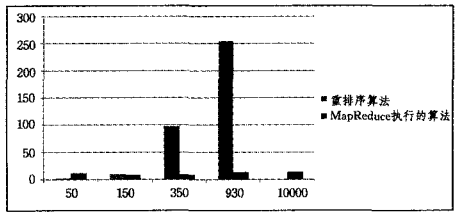

图 3 重排序算法与基于 MapReduce 排序算法的时间比较

除此之外,FAME 框架能够使约 92%左右的冲突得到解决,这是由于框架的排序算法并不能让排序后的规则的动作完全符合该规则涉及的每一个数据包空间片段的动作。但是本文提出的排序算法是以数据包空间片段为排序数据,以片段的动作作为新规则的动作,这就保证了新规则能够接近 100%地解决策略中存在的冲突。

结束语 本文对 FAME 框架进行了改进,改变了原来框架中时间复杂高、根据数据包空间片段来对规则进行排序的算法,采用一种基于 MapReduce 模型的并行化排序算法来直接对数据包空间片段进行排序,并将排序后的片段转化为规则的表示形式,将该片段的约束动作作为该规则的动作。本文提出的算法简化了原框架中的排序,并且直接将数据包空间转化为规则形式来代替原有规则,保证了处理数据包的准确性。

转换后的规则数目理论上会大于原来规则的数目,这对于数据包的匹配来说可能会增加匹配时间,因此本文将来的工作可以研究转换后的规则的合并问题。由于处理后的规则是按序排列的,将来可以针对这种规则的特点制定相对应的数据包匹配算法,来简化本类规则的数据包匹配。

参 考 文 献

[1] Al-Shaer E, Hamed H. Discovery of Policy Anomalies in Distributed Firewalls[C]// IEEE INFOCOM '04. vol. 4, 2004: 2605-2616

[2] Yuan L, Chen H, Mai J, et al. Fireman: A Toolkit for Firewall Modeling and Analysis[C]// Proc. IEEE Symp. Security and Pri-

vacy. 2006:15

[3] Alfaro J, Boulahia-Cuppens N, Cuppens F. Complete Analysis of Configuration Rules to Guarantee Reliable Network Security Policies[J]. Int'l J. Information Security, 2008, 7(2): 103-122

[4] Hari A, Suri S, Parulkar G M. Detecting and resolving packet filter conflicts[C]// INFOCOM. 2000(3): 1203-1212

[5] Fu Z, Wu S F, Huang H, et al. IPSec/VPN security policy: Correctness, conflict detection, and resolution[C]// Proceedings of Policy2001 Workshop. January 2001

[6] Golnabi K, Min R K, Khan L, et al. Analysis of firewall policy rules using data mining techniques[C]// IEEE/IFIP Network Operations and Management Symposium (NOMS 2006). April 2006

[7] Hu H, Ahn G J, Kulkarni K. Detecting and resolving firewall policy anomalies[J]. IEEE Transactions on Dependable and Secure Computing, 2012, 9(3): 318-331

[8] Abedin M, Nessa S, Khan L, et al. Detection and resolution of anomalies in firewall policy rules[C]// DBSEC'06 Proceedings of the 20th IFIP WG 11.3 Working Conference on Data and Applications Security. 2006: 15-29

[9] 田大新,刘衍珩. 数据包过滤规则的快速匹配算法和冲突检测[J]. 计算机研究与发展, 2005, 42(7): 1128-1134

[10] Al-Shaer E, Hamed H, Boutaba R, et al. Conflict Classification and Analysis of Distributed Firewall Policies[J]. IEEE Journal on Selected Areas in Communications, 2005, 23: 2069-2084

[11] Mell P, Scarfone K, Romanosky S. A Complete Guide to the Common Vulnerability Scoring System Version 2.0[K]. Published by FIRST—Forum of Incident Response and Security Teams. June 2007

[12] 张永铮,方滨兴,迟悦,等. 用于评估网络信息系统的风险传播模型[J]. 软件学报, 2007, 18(1): 137-145

[13] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communication of the ACM-50th anniversary issue, 2008, 51(1): 107-113

[14] 李建江,崔健,王聃,等. MapReduce 并行编程模型研究综述[J]. 电子学报, 2011, 39(11): 2635-2642

(上接第 37 页)

如图 5 所示,在一个节点内运行多个进程,随着进程数增加,加速比几乎线性增加。在进程数为 12 时,加速比为10.9,并行效率为 90.8%。

如图 6 所示,在多个节点中,使用多进程测试。进程数从 1,2,4 一直倍增到 256。在 32 进程时,加速比为 28.9,并行效率为 90.3%。在进程数达到 256 时,加速比为 184.6,并行效率为 72.1%。综合图 5 和图 6,软件在超龙一号 X86 部分单节点和多节点运行时,都有很好的加速效果。

对比图 4 和图 6 的加速比,二者的加速效果不同,可能原因是在图 4 中,蛋白质定量程序运行在曙光 6000 龙芯部分,而图 6 中,程序运行在超龙一号 X86 平台,二者的 CPU 频率不同,其他的硬件配置也不同,所以程序运行在这两个平台加速比不同。

如图 7 所示,在多个节点中,使用多进程测试。进程数从 1,2,4,8 到 16。在 16 进程时,加速比为 14.36,并行效率为 89.7%。软件在龙芯部分得到很好的加速效果,并行效率达到与 X86 CPU 相同乃至更高水平。

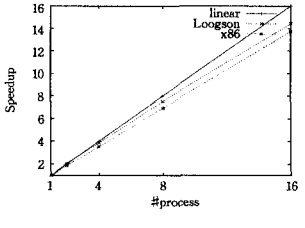

图 7 龙芯平台和 X86 CPU 测试结果对比

4.2 PG-QuantWiz 测试结果

在 X86 部分,MPI+CUDA 版软件 PG-QuantWiz 主要的测试结果如图 8 所示。

PG-QuantWiz 在单 GPU 运行时间为 498.19 秒,结合 PG-QuantWiz 运行在超龙一号 X86 部分单 CPU 的情况,MPI+CUDA 版软件 PG-QuantWiz 在单 GPU 上运行的加速比为 8.1。16 GPU 时间为 35.11 秒,16 个 GPU 对单 GPU 的加速比为 14.18,并行效率为 89%。这表明 MPI+CUDA 版软件 PG-QuantWiz 也有很好的加速效果。

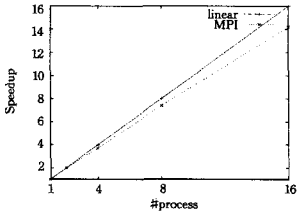

图 8 PG-QuantWiz 测试结果

但是因为测试数据规模有限,在较大的并行环境下,其可扩展性受到影响。因此需要对其进行进一步的优化,包括进一步加大数据测试规模,并根据负载平衡进行数据动态划分。

参 考 文 献

[1] 胡泽林. 基于质谱的高性能蛋白质非标记定量软件设计与性能优化[D]. 2009

[2] MPI: A Message-Passing Interface Standard [OL]. http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html

[3] NVIDA corporation. NVIDIA CUDA C Programming Guide version 4.2[Z]

[4] 费辉. 蛋白质定量和分子动力学模拟若干算法的 GPU 并行设计与实现[D]. 2011

[5] Wang Jing, Zhang Yun-quan, Zhang Xian-yi, et al. QuantWiz: A Parallel Software Package for LC-MS-based Label-free Protein Quantification[C]// Accepted by 2009 International Workshop on Parallel Algorithm and Parallel Software. 2009: 683-687

[6] 费辉,张云泉,王可,等. 基于 GPU 的分子动力学模拟并行化及实现[J]. 计算机科学, 2011, 38(9): 275-278