一种网格环境下的密码计算模型

姜中华[1,2], 林东岱[1]
(1. 中国科学院软件研究所信息安全国家重点实验室, 北京 100080; 2. 中国科学院研究生院, 北京 100049)

摘 要: 由于密码学和信息安全领域的许多问题最终都被转化为一个耗时的计算, 其中许多计算需要利用多台异构的和地理分布的计算机协同, 才能有效完成. 密码算法的设计、分析和应用对于计算环境敏感, 且依赖性较强, 不同类型的算法和算法的不同实现模式对计算环境要求差异很大, 而且到目前为止还不存在一种通用的分布式密码计算模型. 为此, 本文根据密码计算本身的需求, 首先分别分析了密码算法设计、分析和应用的目标和特征, 提出了相应的计算模式, 给出了一种网格环境下的通用密码计算模型. 进而讨论了密码计算任务分割策略, 资源分配和负载平衡问题. 最后给出了网格环境 Globus Toolkit 下的模型构架、实现与实验结果.

关键词: 密码学; 资源分配; 任务分割; 计算网格
中图分类号: TP393 文献标识码: A 文章编号: 0372-2112 (2006) 10-1787-06

A Grid Computing Model for Cryptology

JIANG Zhong-hua[1,2], LIN Dong-dai[1]
(1. *State Key Laboratory of Information Security, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*;
2. *Graduate School of Chinese Academy of Sciences, Beijing* 100049, *China*)

Abstract: Many problems in cryptology and information security fields are eventually converted to time-consuming computations which usually need to be completed over heterogeneous and geographically distributed computers. The analysis, design and application of a cryptographic algorithm usually depend on its concrete computing settings; different algorithms and different implementations of an algorithm need different computing environments. In this paper, the targets and features of different cryptographic computations are firstly analyzed according to their computing requirements; the corresponding computing modes are put forward; and a general purpose cryptographic computing model on grid is given. The task dividing policy, resource allocation and load balance for the model are then discussed. The implementation and the performance evaluation for the model based on Globus Toolkit 4 are further presented.

Key words: cryptology; resource allocation; task dividing; computational grid

1 引言

密码计算的分布式和并行化是信息安全领域中的一个重要问题. 现代密码学基于计算安全性, 即算法安全性与求解某个计算问题等价, 其强度是安全参数的函数. 随着密码理论研究的不断深入和广泛的应用, 面临的计算问题越来越复杂, 这些问题在单台计算资源上通常要花几天、甚至几十天的时间, 或者由于计算能力的限制, 问题就根本不能在有效时间内求解. 从 20 世纪 90 年代开始, 分布式密码计算逐步成为研究热点[1~3].

密码算法的设计、分析和应用对于计算环境敏感且依赖较强, 不同类型的算法和算法的不同实现模式对计算环境要求差异很大. 1987 年, Lenster[4] 研究如何用电子邮件散布程序和处理器间通信, 进行分布式大整数因子分解, 并在一个月内分解了一个 100 位整数. 1990 年 White[5] 研究如何利用计算机病毒散步密码计算程序进行分布式密码计算. 1991 年 Quisquater[6] 提出利用中国式抽彩法进行分布式密码计算思想. 1999 年 Selkirk[7] 研究 TCP/IP 连接和电子邮件通信求解离散对数问题, 连接 16 个国家的 1200 台计算机在 53 天内求解了椭圆曲线挑战 ECCp-97. 2000 年 Asbrink[8] 研究如何用 MPI 机制进行并行因子分解. 另外, 国外一些组织和研究机构为了求解 RSA 实验室和 Certicom 公布的某个 DES、RC6、RSA 或者椭圆曲线求解挑战, 构建专用分布式计算环境, 完成了一些挑战. 如 distributed.net 组织、美国自然科学基金会以及 Internet 爱好者等通过多种方法, 最终解决了 DES-I、DES-II 和 DES-III 挑战, DES-III 只花费 22 小时 15 分钟时间. 尽管国内也取得了

收稿日期: 2005-06-02; 修回日期: 2006-08-20
基金项目: 国家 973 重点基础研究发展规划(No. 2004CB318004); 国家自然科学基金(No. 90204016); 国家 863 高技术研究发展计划(No. 2003AA144030)

不少研究成果,但这方面的高性能计算由于存在跨学科等问题而比较滞后,研究均是给出某一算法的并行化方法.由于密码计算本身的复杂性和多样性,已有研究并没有试图给出一个通用的分布式密码计算模型,实现基于广域网的高性能密码学相关计算.网格[9~11]的目标是实现大规模资源共享,为虚拟组织(VO)[9]提供和协作使用远程高端计算机、数据库、科学仪器、网络和其它资源的基础设施.网格希望屏蔽设备的地理位置和无关细节,为终端用户提供通用的计算能力.网格计算技术已经被广泛的应用到许多领域.然而到目前为止,还没有将网格技术引入到大规模分布式密码计算.鉴于密码计算本身具有自然可分布、节点之间没有消息传递,且通常需要大规模计算资源的特点,非常适合在网格环境中进行.

本文紧紧围绕密码计算和网格计算的特点提出了一种通用密码计算模型.第 2 节分析了密码计算的应用分类和计算类型,并提出了相应的计算模式.第 3 节给出了密码计算任务的分割策略.第 4 节讨论了模型的资源分配和负载平衡.第 5 节给出了模型的构架、实现和实验结果.第 6 节对全文进行了总结.

2 计算模型

通用密码计算模型需要考虑到密码计算的不同类型.同时与大多数分布式计算问题不同的是,密码计算考虑利用大规模分布式计算资源,特别在动态的网格资源环境下,如何保持计算的可扩充性,及资源的有效使用.

2.1 密码计算的分类

密码计算按功能可分为密码算法设计、分析和应用三大类.下面分别对这三类应用对应的计算进行分析.

(1)算法设计.对新密码算法,设计者需要在相同的计算资源条件下,针对不同的安全参数,以及存在的所有分析方法进行分析和测试.为了在有限时间内完成测试,安全参数的取值往往比实际使用的小,因此需要的计算资源数量相对较小,且需要精确统计资源的使用量.

(2)算法分析.密码算法分析和评估,如分组密码的差分、线性、相关密钥分析,哈希函数的碰撞搜索,对密码学依赖的困难问题的研究和求解,密码算法基本结构的分析和比较,伪随机函数的随机性分析,密码学协议的安全性分析,如双方、群组密钥协商协议的分析和证明等.算法分析通常分为两种情况.一种是使用已有的密码分析算法对某种方案在更大安全参数下进行实际计算.另一种情况,对某类密码算法,设计了一种新的分析算法,利用新算法在适当安全参数下,对密码算法进行分析,而且往往需要与已有分析方法进行比较,测试新分析方法的适应性和性能.这类计算往往需要大规模计算能力,且事先无法估计计算时间,其目标是在尽可能短的时间内完成,因此算法关注网格系统的吞吐量,而且要求随时能终止计算.

(3)算法应用.常规密码运算,如加密/解密、签名、哈希运算等.如果应用系统本身没有足够的处理能力(如智能卡和手机等低端设备),或者需要的密码计算能力单台普通计算设备无法满足时,需要在网格环境提供这种能力.它们对资源的需求量往往有限,但是需要满足特定的服务质量(QoS),而且可能需要进行并行处理.

针对密码计算的 3 种应用的各自特点,以及密码学计算与常规计算的区别,我们总结出 3 种计算类型,分别为:概率性大计算(A 类)、确定性大计算(B 类)和普通计算(C 类).(1)A 类计算与常规计算不同的是,密码学中的许多计算(密码分析,如 QS 和 NFS 因子分解算法,并行 Pollard Rho 算法等)属大规模计算,为了有效求解,大都采用概率算法[12].特点是计算量大、数据量小、随机算法、子任务个数和计算量事先无法确定.(2)密码学中的一些计算(密码分析和应用,如对称密码的蛮力攻击、差分分析和线性分析等)计算量大,采用确定性算法.特点是计算量大、数据量小、确定性及计算量事先可以确定,但子任务数量可能是未知的.(3) C 类计算不一定可并行,可能是概率算法,也可能确定性算法(如常规计算和算法设计),但是与前两种类型相比,其特点是计算量相对小.

2.2 密码计算的模式

密码计算具有天然的可分布特点,使得密码计算非常适合在网格计算环境下快速求解和计算:(1)计算代码短,迭代执行;(2)通信量很少,计算节点通常与服务器只有少量交互;(3)结果容易重构;(4)数据量小,大多数计算只需少量的明/密文对等信息;(5)计算量大;(6)计算代码可不断优化;(7)可分布式和并行计算[7].

定义 **1** 负责任务分割和拼接的网格节点称为控制节点(NC);负责实际计算的网格节点称为计算节点(NA).注意 NC 和 NA 可为同一网格节点.

定义 **2** T 表示一个密码计算任务.DA、CA 和 AA 分别对应任务的任务分割、拼接和计算算法.

针对 2.1 节的三类密码计算,我们分别给出在网格中的计算模式.

1. 概率性大计算(A 类)的计算模式

A 类计算需要大量 NA,且 NC 的负载不能过重.由于这类计算的子任务数据是相同的,因此只需在 NC 维护一份子任务数据,不需要存储和检索子任务列表.但要求 NA 不断迭代计算,当 NC 得到整个计算的结果时,需要一种机制通知 NA 结束计算.我们采取如图 1 所示的方法(其中* 表示迭代操作):如有空闲 NA,NC 就利用子任务备份向该 NA 提交该子任务.每个 NA 接收到子任务后,先产生适当的随机数,然后开始计算,当子任务计算完成后,如果得到部分有效结果,则提交给控制节点 NC,否则不与 NC 交互.然后 NA 重新产生随机数,重复同一子任务的计算.NC 除了给新 NA 提交同一计算任务外,就是收集并拼接子任务的计算结果.一旦 NC 接收到足够的子任务结果,得到最终结果后,NC 发送广播给所有参与计算的 NA,使 NA 终止子任务计算.

与普通计算相比,该模式优点是:NC 不需要存储和检索子任务列表,甚至不需要维护 NA 列表,因此 NC 的负载轻,只需提交子任务和拼接子任务结果,减少了 NC 和 NA 间的通信量,非常适合在网格环境中的大规模计算.大数因子分解 NFS 算法和 QS 算法[2,8]、椭圆曲线离散对数的 Pollard Rho 算法[7]都可使用该模式并行计算.

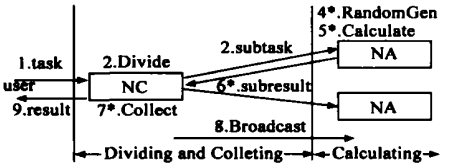

图 1　概率性大计算的计算模式

例 1: 并行计算椭圆曲线(EC)离散对数. 已知 EC 上两点 P、Q 及关系 $Q=dP$, 求整数 d. Pollard Rho 算法的并行计算过程如下: NC 首先在 EC 上选择一个均匀分布点集 D 的特征 F, 然后将(P, Q, F)发送给空闲的 NA. 接收到请求之后, NA 生成随机整数对(a_0, b_0), 并计算 $X_0=a_0P+b_0Q$. 然后再使用固定的迭代函数计算点列 $X_i=a_iP+b_iQ$. 当 $X_i\in D$ 时, NA 将(X_i, a_i, b_i)传给 NC, 如果 NC 事先已经接收到 X_i 另一个对(X_i, a'_i, b'_i), 则利用扩展的 GCD 算法容易求得 $d=(b'_i-b_i)\cdot(a_i-a'_i)^{-1}$, 否则 NC 保存该三元组, NA 继续迭代. 求得 d 后, NC 给所有 NA 发送一个广播, 然后 NA 结束计算.

2. 确定性大计算(B 类)的计算模式

NC 必须维护子任务列表$\{(T_i, NA_i)\}$, 否则计算无法容错. 由于子任务数据各不相同, 每个子任务都可能包含计算结果的一部分, 或者是整个计算结果. 在计算过程中, 往往由于网络失败或者 NA 失败, 某些 NA 可能无法将其结果传回 NC, 这时 NC 即使得到其它所有子任务的结果, 也可能无法求解整个计算的结果.

确定性算法求解大密码计算问题可能导致计算失败. 当 NA 数量很大(例如两亿)时, 尤其在网格环境中, 很可能由于 NC 维护了太大的列表$\{(T_i, NA_i)\}$, 而耗尽 NC 的所有资源, 最终导致计算失败. 我们用如下两种方法之一来解决.

对于穷举类算法, 借鉴文献[6]的思想, 我们将其转化为概率算法, 称为概率穷举算法. 对其它类型的确定性算法, 我们采用增加 NC 的数量来解决. 下面我们给出概率穷举算法:

算法 **1**　(概率穷举算法)NC 不维护子任务列表$\{(T_i, NA_i)\}$. NC 首先向每个空闲 NA 发送明文、密文和密钥空间对(M, C, K), 每个 NA 接收到 NC 的任务数据后, 生成一个随机数 $R\in[0, K)$, 根据 NA 的计算能力, 设置一个适当的区间长度 L, 使$[R, R+L)\subset[0, K)$, 然后在该区间内穷举, 如果该 NA 在区间$[R, R+L)$找到密钥 k, 则将结果传给 NC, 否则再生成新的随机值 R', 继续对产生的新区间$[R', R'+L)$进行穷举. 如果 NC 接收到密钥 k 后, 判断 $C=E_k(M)$, 如果成立, 则广播"计算完成"信息, NA 接收到广播后, 立即停止计算.

概率穷举算法的优点是使 NC 的负担很轻, 不需要存储和检索子区间列表$\{T_i, NA_i\}$, 而且 NC 和 NA 之间的通信减少到最小, 几乎为零.

例 2: DES 和 AES 的穷举都可以采用概率穷举模式, 给定哈希函数 MD4、MD5 和 SHA1 的差分路径[14], 也可以采用类似算法 1 的概率穷举方法寻找碰撞.

如果概率穷举算法能够实用, 必须具备如下两个条件: (1)整个穷举密钥的数量应该与确定性穷举方式相当; (2)概率算法的失效概率要足够低. 满足第 1 个条件保证不会浪费太多计算能力; 满足第 2 个条件保证绝大多数情况下, 计算能够正常完成.

定理 **1**　算法 **1** 随机搜索时, 尝试次数期望值近似于密钥空间大小除以参与概率穷举的 NA 数量.

证明: 假定 K 是密钥空间的大小(对 DES 算法 $K=2^{56}$), W 表示并行工作 NA 的数量, 我们证明 W 个 NA 并行穷举时, 每个 NA 穷举密钥数量的数学期望为 K/W. 假定每个 NA 都(均匀的)搜索 i 个不同密钥后, 还没有找到正确密钥, 则其概率为 z^i, 其中 $z=((K-1)/K)^W$.

令 $P(i)$表示在第 i 次搜索时至少有一个 NA 找到密钥, 但前 $i-1$ 次都没有找到的概率. 则

$$P(i)=(1-z)\times z^{i-1}$$

使用算法 1 找到密钥的穷举密钥次数的数学期望为:

$$E(i)=\sum_{i=1}^{\infty} i\times P(i)=(1-z)\sum_{i=0}^{\infty}(i+1)z^i$$

令 $f(z)=\sum_{i=0}^{k}(i+1)z^i$, 为了计算 $f(z)$, 我们计算$(1-z)f(z)$, 有:

$$\begin{aligned}(1-z)f(z)&=1+z+\cdots+z^k-(k+1)z^{k+1}\\&=(1-z^{k+1})/(1-z)-(k+1)z^{k+1}\end{aligned}\qquad(1)$$

对式(1)两边取极限, 可得$\lim_{k\to\infty} f(z)=(1-z)^{-2}$.

从而 $E(i)=1/(1-z)$, 将 z 按幂展开, 由于 $W\ll K$, 因此有 $z\approx 1-W/K$, 即 $E(i)=K/W$.

定理 **2**　算法 **1** 失效的概率足够低.

证明: 为讨论最坏情况, 我们计算每个 NA 穷举 $K/W\cdot f(K)$个密钥后, 还没找到密钥的概率. 其中 $f(K)$是 K 的函数. 使用定理 1 计算的概率, 可得概率为: $(1-1/K)^{K\cdot f(K)}$. 通常情况下 K 很大, 因此$(1-1/K)^{K\cdot f(K)}\approx e^{-f(K)}$. 当 $f(K)=\ln K$ 时, 算法 1 失败的概率为 K^{-1}.

事实上算法 1 将 B 类的一些计算转化为 A 类计算. 对不能用算法 1 求解的确定性密码计算, 需要在 NC 端存储和检索子任务列表$\{(T_i, NA_i)\}$, 为了保证计算的可扩充性, 我们采取多个 NC 协作分割和拼接的策略.

3. 普通计算(C 类)的计算模式

由于 C 类计算的规模相对小, 因此可采用常规方式处理, 即 NC 存储和检索子任务列表, 当 NC 接收到某子任务的结果后, 进行结果拼接, 之后从列表中删除对应的子任务, 如果该子任务计算失败, 则利用存储在子任务列表中数据, 再次提交到其它空闲的 NA. 对安全参数相对小的计算都可采用这种方式, 且适合大多数除密码分析之外的计算, 但 NC 有维护子任务列表的负担.

3　计算任务分割策略

任务分割是分布式计算必须面对的基本问题. 理想情况是给出通用的分割算法, 使任务分割自动化. 鉴于密码计算的多样性, 不同算法的差异很大, 因此不存在通用分割算法. 许多计算事先无法精确估计计算量. 考虑到网格资源的动态性, 事先无法预知可用的网格资源数量. 且确定性大计算的子任务数量通常很大, 难以存储事先分割的所有子任务. 为此, 我们试图给出一种动态任务分割策略, 使之较好适应三种计算类型密码计算. 有些密码计算需要聚合多种已有的密码算法,

才能得出计算结果. 因此需采用递归的任务分割策略. 同时密码计算具有任务分割计算量很小, 而任务计算计算量通常很大的特点.

我们的策略: 把计算任务分为原子任务(AT) 和非原子任务(DT) 两大类. AT 不需再分, 直接计算; 而 DT 则需要被(递归) 分割成更小任务(AT 或 DT), 直到分割成 AT. 任务结果拼接过程是分割的逆过程.

定义 **3** 任务分割树(TT) 是指通过该策略, 将用户提交 $[K_1, K_2)$ 的任务(根任务, RT) 递归地分割以原子任务为树叶, 以非原子任务为树枝的树.

例 3: 利用 8 轮 DES 差分分析[13] 和 DES 穷举结合求 DES 算法的 56 比特密钥. 任务数据包括随机密钥对集合 $S = \{(C_1, C_2)\}$ 以及一明密文对 (M, C) 其中 $|S| = 150000$, NC 首先对 S 预处理, 得到 $S' = \{(C'_1, C'_2)\} \subset S$, 其中 $|S'| \approx 20000$, 然后利用 S' 进行并行差分分析, 求得 30 比特 DES 密钥, 再根据 (M, C), 进行并行穷举得到剩余 26 比特密钥. 该计算的 TT 有 3 层, 第 1 层是该任务本身, 第 2 层是 DES 差分和 DES 穷举两个串行任务, 第 3 层是 DES 差分的可并行原子任务, 以及 DES 穷举的可并行原子任务.

该分割策略具有灵活性, 使得可在计算过程中, 根据网格资源和计算任务的状态改变计算模式. 另外任务分割与具体分割算法和任务粒度有关, 如果分割算法得到改进, 则原来无法再分的任务可能由 DT 变成 AT, TT 树的深度会随之增加.

例 4: 假定 NC 进行一个 DES 密钥区间穷举 $RT = (M, C, [K_1, K_2))$. NC 给每个空闲 NA 提交 $[K_1, K_2)$ 的一个子区间, NA 进行确定性计算. 当某个时刻, 可能 NC 的资源不足, 或根任务太多, 使之无法存储和检索 RT 的子区间列表, 假定 RT 已穷举了 $[K_1, K'_2) \subset [K_1, K_2)$, 则 NC 可删除 RT 的子区间列表, 然后产生新子任务 $(M, C, [K'_2, K_2))$, 使用概率穷举算法计算, 当该子任务计算完成后, 将结果提交给 RT 的拼接算法, 最后 NC 结束计算.

4 资源分配和负载均衡

资源的异构性和动态性是网格的重要特征, 虽然有许多研究成果, 然而由于网格计算的复杂性, 网格资源分配仍是一个挑战性问题. 密码计算涉及的算法设计、分析和应用的计算目标和对资源的要求不同, 而且概率算法的子任务数量不确定等, 研究表明, 对这类计算的最优资源分配是 NPC 问题.

在网格系统中, 同时存在多个 RT, 多个 NC 和 NA. 根据计算的类型, 我们采用不同的优化策略. 为了对资源的限制和任务需求进行描述, 我们采用 ClassAD 表示方法[15], 为网格资源以及计算子任务创建资源描述.

根据三类密码计算的特点, 定义了三种目标函数, 通过线性规划方法[17], 分别得出最优的分配方案: (1) 资源代价, 密码算法设计需要在固定的资源上对不同的分析方法进行测试, 目标是匹配一组代价最低的资源; (2) 吞吐量, 密码分析需要在尽可能短的时间内完成, 且计算规模大, 因此需最大化资源的吞吐量; (3) 负载平衡, 对密码应用的计算, 用户希望得到满意的 QoS, 需要有较高的计算可靠性.

定义 **4** 设 R 是一个网格资源集合, J 是需要匹配的作业(子任务) 集合, P 表示 R 的每一个资源的优先级集合, T 表示资源类型集合. $N = \{n_{r,t}\}$ 表示资源 r 的类型为 t 的能力集合. $U = \{u_{(j,t)}\}$ 表示资源需求集合, $u_{(j,t)}$ 表示作业 j (子任务) 对类型 t 的资源的需求量. $X = \{x_{(j,r)}\}$ 中, 如果子任务 j 与资源 r 进行了匹配, $x_{(j,r)} = 1$, 否则取值 0. $Z = \{z_j\}$ 中, 如果作业 j 需要的所有资源都匹配成功, 则 $z_j = 1$, 否则取 0.

下面给出这些目标函数:

1. 资源代价(Cost)

该目标函数 f 定义为匹配成功后所使用资源的数量, 希望在尽可能少的资源上运行一批作业, 以降低费用, f 应取最小值. 对 $\forall r \in R$, 引入新变量 $s_r \in \mathbf{Z}_2$, 且定义不等式: $\sum_{j \in J} x(j, r) \geqslant s_r$ 和 $\sum_{j \in J} x_{(j,r)} \leqslant s_r * |J|$. r 没有分配给任何作业时, $s_r = 0$, 否则为 1. $\sum_{j \in J} x(j, r)$ 表示分配给 r 的作业数. 则 $f = \sum_{r \in R} s_r$.

2. 吞吐量(Throughput)

该目标函数 f 定义了一次匹配过程中获得资源的作业数. 当 f 取最大值时, 结果将找到一个匹配尽可能多作业的方案. $f = \sum_{j \in J} z_j$ 反映吞吐量. 若考虑作业优先级 p_j, 则 $f = \sum_{j \in J} p_j z_j$.

3. 负载平衡(Load Balance)

该目标函数 f 表示负载最重的资源上空闲能力的百分比. 使 f 取最大值, 结果将作业从负载最重的资源移动到负载较轻的资源上, 从而达到某种程度的负载平衡. 对 $\forall r \in R$, 设 $g_{(r,t)} \in [0, 1]$ 表示在资源 r 上能力类型 t 已使用的百分比. 对 $\forall r \in R$, $t \in T$, 有 $\sum_{j \in J} u_{(j,t)} * x_{(j,r)} = n_{(r,t)} * g_{(r,t)}$ 及 $f \leqslant 1 - g_{(r,t)}$ 成立.

根据作业的需求和资源的限制将 VO 划分成子 VO, 再构造相应的目标函数, 使用网络气象服务(Network Weather Service)[16] 收集资源的动态属性, 获得网格资源的实时负载信息, 便于再次利用目标函数进行优化分配.

5 系统构架和实现

5.1 构架

模型构架在网格工具 Globus Toolkit[10] 和其服务的基础之上, 是一个通用的、开放的、可扩展和高效的密码计算环境. 为此, 模型需要密切关注任务分割、任务计算、执行管理以及资源管理(计算引擎、计算代码和计算数据) 等问题, 它们是通过协作几个基本的和专用的网格服务解决的[18], 如图 2 所示.

网格服务分为两层: 核心层和高层, 还包括元数据仓库. 核心层主要进行资源管理, 包括 CDEDS 和 RAMS 服务, 该层

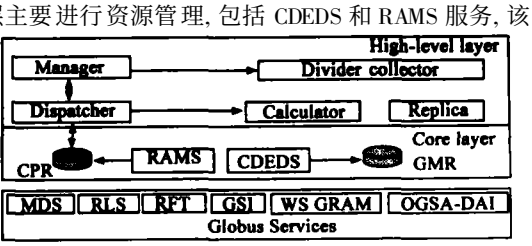

图 2 密码计算模型的构架

也根据计算需求和可用网格资源协调密码计算的执行. 高层描述、控制和执行并行和分布式密码计算任务, 且协调网格节点间的代码和计算数据复制, 包括 Dividercollector、Manager、Dispatcher 和 Calculator 服务.

CDEDS 扩充了 Globus MDS4 服务, 用于维护网格可用的所有计算代码、计算数据和计算引擎. RAMS 自动在一个动态的执行计划(EP)和可用资源间建立优化映射; 密码执行计划仓库(CPR)用来备份 EP; Dividercollector 自动装载和使用任务分割和拼接算法后, 进行任务分割和任务拼接; Manager 管理 Dividercollector; Calculator 负责原子任务计算; Replica 实现在远程站点和本地站点间复制计算代码、计算引擎和大数据集, 它依赖于 CDEDS; Dispatcher 不仅是子任务调度器, 且控制网格节点间的交互; RAMS 根据可用的资源和 EP 中子任务的资源需求, 决定将计算任务与哪一个网格节点相匹配.

CDDES 管理的元数据主要用来描述对象的特征和位置信息, 包括如下类型的对象: 计算任务和计算结果; 用于分割、拼接、计算任务的代码以及计算引擎. 元数据表示为 XML 格式, 被存储在网格元数据仓库 (GMR) 中. 该服务也用于对计算数据的搜索和访问, 查找注册过的代码和计算引擎.

5.2 实现

为了重用任务分割、拼接和计算算法(DA、CA 及 AA), 我们将它们包装成插件 Plug in, 并实现相应的接口, 经 CDDES 将其元数据存储在 GMR 中, 根据需要, 网格节点可通过 Replica 服务并查询 GMR, 复制这些插件.

NA 利用 Calculator 计算原子任务. NC 通过 Dividercollector、Manager 和 Dispatcher 对多个 RT 递归分割和拼接. 多个 NC 可能协作完成 B 类计算. 每个子任务数据包括(TaskName, RootName, ParentName, ProblemType, AppType, ComputingType, TaskData). Manager 根据 ProblemType 匹配对应的 Plug in, 根据 TaskName 进行拼接, 根据 ParentName 进行递归结果拼接, 根据 RootName 给所有 NA 发送广播, 根据 AppType 决定资源优化目标. 根据 ComputingType 决定计算模式(A、B 或 C 类计算, 或者模式的组合), TaskData 是实际计算数据.

我们通过 Globus Toolkit 的通知/预定机制模拟广播的发送. 得到子任务的 NA 向 NC 预定"计算终止"消息. 为了简化消息传递, 在服务之间和网格节点间定义了独立于底层协议的、基于 XML 的密码计算协议 CCCP.

为简化插件开发, 可以集成已有的密码学计算函数库和计算引擎, 目前已集成了库 Crypto++ 和引擎 Maple. Manager 和 Calculator 可通过 globusrun 程序或 Java JNI 调用这些引擎. 另外我们提供了一个门户站点, 用户可通过该站点方便与网格系统交互.

5.3 性能评价

表 1 给出了由多台普通 P4 2.2GHz CPU、256M RAM 的 PC 机组成的网格环境中进行密码计算的实验结果. D1 和 D2 是对 DES 密钥子区间[0, 0×C00, 000H)并行确定性蛮力攻击结果. E42 和 E43 用并行 Pollard Rho 算法对 42 和 43 比特素域上 EC 的计算结果, DD 给出了 8 轮 DES 差分并穷举计算 56 比特密钥的结果.

表 1　网格环境下密码计算实验数据

No	Serial Time	Speed-up		
		2NA	3NA	4NA
D1	472.04	1.996	2.98	3.92
D2	472.04	1.92	2.84	3.78
E42	245.65	1.91	2.78	3.74
E43	357.62	1.90	2.81	3.75
DD	1973.2	1.94	2.80	3.60

结果表明, 加速比随 NA 数保持稳定增长, 且接近线性. 对两次 EC 计算的加速比虽接近, 但存在随机波动, 如当两个 NA 时, 42 和 43 比特 EC 的加速比分别为 1.91 和 1.90, 而 3 个 NA 的加速比为 2.78 和 2.81, 原因是概率算法导致碰撞具有随机性. 对于两次 DES 蛮力攻击的差别有两点: (1) D1 比 D2 的任务划分粒度更细, 会增加并行开销; (2) D2 用两个 NC, 而 D1 只用一个 NC. 注意本实验在小规模网格环境中进行, 因此有些实验结果可能与大规模环境有所差别.

6 结论

随着网格基础设施的发展, 它支持的工具和相应的应用越来越多样化. 本文紧紧围绕密码计算和网格计算环境的各自特点, 提出了一个网格环境下的密码计算通用模型. 首先讨论了三类密码学应用的特点, 然后抽象出三类密码计算类型, 针对每种类型, 给出了计算模式. 为了有效实施这三类密码计算, 提出了递归的任务分割策略. 应密码计算应用的不同需求, 利用线性规划来优化对网格资源的分配, 针对密码算法设计、算法分析和普通计算分别给出了三种优化目标. 通过最大化或者最小化目标函数, 即可得到优化的资源分配解. 最后给出了实现模型的网格服务构架、实现和实验结果. 结果表明, 提出的模型适合在网格环境下进行高性能计算.

参考文献:

[1] van Oorschot P C, Wiener M J. Parallel collision search with application to hash functions and discrete logarithms[A]. In Proc the 2nd ACM Conference on Computer and Communications Security[C]. ACM Press, New York, 1994. 210- 218.

[2] Brandon Dixon, Arjen K. Lenstra. Factoring integers using SIMD sieves[A]. In Proc EUROCRYPT'93[C]. LNCS 0765. Heidelberg: Springer Verlag, 1994. 28- 39.

[3] Dan Page. Parallel Solution of Sparse Linear Systems Defined Over GF(p)[R]. Technical Report CSTR-05-003, University of Bristol, 2004. 11.

[4] Lenstra A, Manasse M. Factoring by electronic mail[A]. In Proc EUROCRYPT'89[C]. LNCS 434. Heidelberg: Springer-Verlag, 1990. 355- 371.

[5] White S R. Covert distributed processing with computer viruses [A]. In Proc CRYPTO'89[C]. LNCS 435. Heidelberg: Springer Verlag. 1990. 616- 619.

[6] Quisquater J J, Desmedt Y G. Chinese lotto as an exhaustive code-breaking machine[J]. Computer. 1991, 24(11): 14- 22.

[7] Selkirk A P L, Escott A E. Distributed computing attacks on cryptographic systems[J]. BT Technology Journal. 1999, 17(2): 69- 73.

[8] Asbrink O, Brynielsson J. Factoring Large Integers Using Parallel Quadratic Sieve[R]. Sweden: Royal Institute of technology, 2000.

[9] Foster I, Kesselman C, Tuecke S. The anatomy of the grid: Enabling scalable virtual organization[J]. International J Supercomputer Applications, 2001, 15(3): 200-222.

[10] Foster I, Globus toolkit version 4: Software for service-oriented systems[A]. In Proc NPC'05[C]. LNCS 3779, Heidelberg: Springer-Verlag, 2005. 2-13.

[11] Foster I, Kesselman C, Nick J, Tuecke S. Grid services for distributed system integration[J]. IEEE Computer, 2002, 35(6): 37-46.

[12] Rosario Gennaro. Randomness in cryptography[J]. IEEE Security and Privacy, 2006, 4(2): 64-67.

[13] Biham E, Shamir A. Differential cryptanalysis of DES-like cryptosystems[J]. Journal of Cryptology, 1991, 4(1): 3-27.

[14] Xiaoyun Wang, Hongbo Yu. How to break MD5 and other hash functions[A]. In Proc EUROCRYPT'05[C]. LNCS 3494, Heidelberg: Springer-Verlag, 2005. 19-35.

[15] Raman R, Livny M, Solomon M. Policy driven heterogeneous resource co-allocation with gangmatching[A]. In Proc IEEE HPDC'03[C]. Washington: IEEE Computer Society. 2003. 80-89.

[16] Wolski R. Experiences with predicting resource performance on-line in computational grid settings[J]. ACM SIGMETRICS Performance Evaluation Review, 2003, 30(4): 41-49.

[17] Wolsey L A. Mixed integer programming formulations for production planning and scheduling problems[A]. Invited talk at ISMP'85[C]. Cambridge: MIT Press, 1985.

[18] Jiang Z, Lin D, Xu L, Lin L. Integrating grid with cryptographic computing[A]. In Proc ISPEC'06[C]. LNCS 3903. Heidelberg: Springer-Verlag, 2006. 321-331.

作者简介:

姜中华 男,1972 年 3 月出生于湖北省襄樊市,现为中国科学院软件研究所博士研究生,主要研究领域为密码理论与技术、网格计算.

E-mail: jzh@is.iscas.ac.cn

林东岱 男,1964 年 4 月出生于山东省聊城市,中国科学院软件研究所研究员,博士生导师,主要从事密码理论、安全协议、网格计算、符号计算与软件设计方面的研究工作.

E-mail: ddlin@is.iscas.ac.cn