XEL - 基于区块链的高度并行网格计算平台即分布式超级计算机

XEL顶层设计解析

原文地址：https://github.com/xel-software/xel-white-paper/blob/master/Whitepaper_XEL_Draft_001.md

翻译：Yijun Huang 邮箱：huangyijun@topgems.co

概述

在这里我们要讨论的是一个基于区块链的去中心化网格计算平台 - XEL，在此平台上你可以轻松获得难以想象的巨大算力。在过去，只有少数科研类项目才能获得类似于超级计算机的巨大算力，但是现在通过XEL管理分布于全世界的计算节点（个人计算机），每个人都可以轻松获得所需的算力。在这之前，某些科研项目需要大量志愿者捐献个人计算机的算力才能展开研究，而现在只需要给计算任务设定适当的加密货币报酬，以此鼓励用户出售算力来获取回报就能开展相应的研究工作。同样在过去志愿者对于需要捐献算力的科研项目是非常挑剔的，只有少数像 在家搜寻外星智慧[2] 这种利他主义项目能激起大量志愿者的捐献兴趣，而不用浪费大量时间跟志愿者纠结这个项目到底有多少科研价值。现在通过引入加密货币做为提供算力的报酬，在XEL平台上，不管是科研项目还是*非科研项目*都能够轻松获取所需的算力。

关键词： 多通道数据流架构，区块链，分布式账本技术，性能

1. 介绍

这个世界上的绝大部分计算资源掌握在数以亿计的普通用户手中，而不是在超级计算机中心或者实验室里。1996年 互联网梅森素数大搜索[3] 第一次提出了 志愿者计算[6] 这个概念并包含一个使用这些资源进行以前不可行的科学计算的范例。分布式计算的潜力很快被其它项目注意到并采用了，比如 在家搜寻外星智慧[2] ，该项目于1999年启动，致力于通过分析来自天空不同部分的无线电信号来搜索外星生命。在2004年，伯克利开放式网络计算平台[1] 项目从 在家搜寻外星智慧[2] 项目中脱颖而出，使得该技术可供公众使用，并且几乎可以通过分布式计算来辅助任何项目。这些项目普遍都拥有 “高度并行计算”[5] 的需求，即将计算任务分解成多个较小的独立子任务，然后将这些子任务分配给不同的计算实体并行执行。例如在 在家搜寻外星智慧[2] 项目中，每个计算实体只分析天空的一小部分，可以跟其它计算实体并行执行。而事实上任何 高度并行计算[5] 的需求都可以以这种方式来完成，包括进化计算，元启发式算法，粒子物理模拟，数值天气预测，CNF求解和加密货币挖掘等等。

然而 志愿者计算[6] 依赖于无私的志愿者，而这些志愿者更倾向于把算力捐献给科研类项目。此外，这些志愿者对于被捐献算力的项目都非常挑剔，项目必须具有一定的科学影响或者能足够引起志愿者的兴趣才行，这就意味着大量需要巨大算力的非科研项目得不到支持。在本文中我们解决了这个问题，并提出一种基于区块链技术构建的分布式计算系统，通过加密货币支付奖励用户，以此来和科研项目共享计算资源。此系统为更广泛的公众创建了一个非利他奖励机制，以此鼓励公众提供计算资源并吸引科学界之外的用户。这使得本系统不仅仅可以用于科学上有意义的计算，还可以为诸如企业界的高度密集计算等提供服务。在本文档的其余部分，我们将从用户的角度详细描述本系统端到端工作流程。

2. 概述

我们的目标是创建一个开源的区块链驱动的中间件系统，用分布式计算解决高度并行计算的问题。项目的重点在于系统的可持续发展，即建立一个不依然于任何人的平台，即没有需要维护的中心化基础设施，更不需要持续的资金投入。我们相信区块链技术在实现这一目标上具有巨大的潜力，它高度冗余的分布式设计使得它异常坚固，也使我们开始重新思考设计整个系统的方式。总而言之，我们使用分布式账本技术创建了一个“更好”的 伯克利开放式网络计算平台[1]。它的网络可任意扩展、高度冗余，并为每一位参与者创建了一个开放，公平和自由的使用环境。而这一切都不需要任何权威的中央机构进行维护。建立审查制度并且以公平的市场价格垄断一批计算资源，用于支持某些偏爱的项目，这在集中式系统环境中是可行的，但在现在的分布式系统环境下则完全没有必要了。

3. 术语

在本文中，"科学家"用于泛指那些寻求使用网络中可利用的计算资源以解决复杂计算工作的人。 而将提供计算资源用于交换加密货币报酬的人称为"矿工"。

在XEL中，矿工可以通过两种方式获得加密货币报酬：为解决问题而持续的工作或者找到问题的答案。矿工们通过持续提交符合特定标准的部分（但不正确）解决方案而获得奖励。这种工作方式被称为"工作量证明[7] 提交"，而潜在的挑战则被称为"工作量证明[7]"。本文中的“赏金”则是为找到问题答案的矿工准备的。这两种报酬分别称为"工作量证明[7] 奖励"和"赏金奖励"。具体细节请参考如何提交你的XEL任务。

4. 端到端工作流程

4.1 创建工作

科学家发布解决问题所需的逻辑代码，同时标明赏金以鼓励大家参与解决问题，这通常是一个工作流程的开始方式。更准确的说，科学家们用 ePL[8] 来编写他们的算法，这是一种专门为XEL设计的DSL(领域特定语言)，它具有高度并行的特点，并可以自由定义找到问题解决方案的标准。有些人可能会担心加入一种新的编程语言会降低XEL的接受度，但我们想指出的是整个IDE(集成开发环境)是非常易用的，它允许科学家们使用自己熟悉的语言（如Python, C, Java）进行编码，而IDE则会自动将代码转换为符合要求

的ePL代码。

上文曾说过，找到问题答案的矿工会获得“赏金奖励”，而这些奖励是科学家们通过XEL虚拟货币支付的。赏金是科学家在创建工作任务时设定的，赏金的额度则受"市场化经济"的影响。因为随着整个算力市场的扩大，科学家们需要为了获取足够的算力而彼此竞争，如果奖金额度太低可能就无法吸引足够的矿工提供算力。

"赏金"是激励矿工们提供算力完成计算任务的主要方式，但是这里存在一个风险：有些计算任务可能根本没有答案，而矿工们也就永远无法获得赏金。这种情况可能来自于科学家的恶意行为，也可能是计算任务的代码出了问题。为了最大程度降低这种风险，科学家们必须要提供额外的"工作量证明奖励"。"工作量证明"被定义为难度适当的任务，但结果确非常容易被验证，同时为了确保矿工的积极性，不管任务有多么难或者多么容易都有一定几率被随机发现。而且"工作量证明奖励"应该不低于矿工们的平均电力成本，以此来减轻矿工们的担忧。

4.2 获取赏金

那么什么是赏金任务呢？可以解释为科学家根据自己的逻辑编写一段程序，该程序接受 伪随机数[9] 输入，并且验证该输入是否构成"赏金奖励"、"工作量证明奖励"或者没有任何奖励。以 货郎担问题[4] 问题举例，伪随机数可以理解为一种特定的解决方案 - 从开始到目的地的哈密顿图[10]，而科学家的程序则用来验证该方案是否低于特定阈值的总成本。

矿工们为了赏金不停的提供算力产生伪随机数，直到找到满足赏金条件的伪随机数为止。这些伪随机数不是完全由矿工们随机生成的，而是由一个叫做"个性化整数"的功能代码帮助生成。"个性化整数"会挑选计算节点公开的和任务指定的某些数值，比如矿工们的公开密钥，任务的ID，任务所在区块的区块ID或者矿工们随机挑选的一些公开数值。这样做是为了给每一个计算结果-伪随机数提供独特的数据id，防止矿工的计算成果被其它人偷窃。如果某个节点拦截了一个计算结果，并且提交占为己有，由于公开密钥的不同则会产生不同的伪随机数，而该伪随机数可能会不满足任务指定的阈值。

当矿工找到赏金任务的解决方案时，他们会先提交某些用于生成伪随机数和任务作者定义计算结果的数值。在网络上的其它矿工确认这个解决方案后，就可以会获得科学家提供的赏金。

4.3 获取"工作量证明奖励"

运行科学家的代码检查一个伪随机数是否满足赏金任务之后，几乎不需要消耗额外的算力即可生成工作量证明方案。在每次运行完任务代码后即会生成一个工作量证明方案 - 由MD5算法加密的哈希值，并与网络设定的工作量证明目标哈希值进行比较，如果生成的哈希值小于目标哈希值则可获得"工作量证明奖励"。跟发现赏金解决方案一样，在发现工作量证明方案后，方案和计算结果会立即被提交。网络中的每个区块都会重新生成一次工作量证明目标哈希值，目标是对于所有正在运行的任务，平均每个区块获得10个（最多25个）工作量证明奖励。也即是说，如果一个区块生成的时间是60秒，网络试图将所有任务的奖励收敛到每分钟10个工作量证明奖励。以赏金的情况类似，只要矿工提交了可以被验证为正确的工作量证明方案就可以获得工作量证明奖励。

4.4 验证提交结果

验证提交结果是整个系统最为关键的一环。然而，虽然执行验证步骤不可或缺，但是许多计算可能过于复杂且耗时，以至于无法在整个网络中及时执行。虽然验证过程可以在不需要人为干预的情况下长时间运行，但是无法保证网络中的每个计算节点都能够在合理的时间内完成验证。因此，对于验证的复杂程度有一个非常严格的要求。如果科学家碰巧有一个极其复杂的算法逻辑，那么他可以选择提供一个替代的简化验证逻辑，只检查解决方案中的一些关键变量来确保结果有效。

在验证"赏金奖励"和"工作量证明奖励"解决方案的环节中，网络必须确保方案确实是来自于提交的节点。这一部分很简单：提交用于验证的伪随机数是由矿工之前提交的一些公开数值（公开密钥，任务的ID，任务所在区块的区块ID。。。）衍生出来的。由于这些数值中包含矿工预先声明的公开密钥，所以只需验证方案中的公开密钥是否与提交解决方案矿工的公开密钥相匹配即可。但在工作量证明方案的情况中，我们需要额外确保计算节点确实执行了计算代码以生成解决方案，而不是仅仅不停搜寻比目标哈希值更小的数值。这可以通过要求工作量证明解决方案提供和赏金解决方案相同的数据，以及包括在运行任务时生成的MD5哈希值来实现。因为验证节点具有验证赏金方案和工作量证明方案的所有数据，所以它们可以简单的运行上述的赏金验证逻辑然后生成MD5算法加密的哈希输出。这就允许验证节点快速的验证提交节点是否有效，以及提交的工作量证明方案是否由运行实际任务所产生。

我们认为，拟议的验证步骤是一个坚实可靠的解决方案，可以阻止预期会遭遇的最常见类型的攻击。

5. 自定义DSL - ePL(领域特定语言)的优势

5.1 兼容绝大部分平台

虽然预计提供算力的大部分平台都是使用典型的x86和x86-64 CPU提供算力，但我们还是致力于设计兼容任何平台的系统，比如运行OpenCL或者CUDA的GPU，以及使用逻辑门操作的FPGA-现场可编程门阵列。ePL使用非常通用的运算符集，提供自上而下的高级功能并以平台无关的方式来描述算法。矿工们可以将ePL转换为某特定平台的代码，并在本地编译它以满足目标平台体系架构。

5.2 保持稳定性

ePL的主要关注点是确保所有任务能及时正常终止，并且不会再将任何线程发布到矿工的平台。

ePL包含一个预处理器，该预处理器用于检查提交的任务是否符合标准的表达方式，同时还包含一个运行时模块用于防止程序崩溃。由预处理器识别的包含堆栈溢出，除于零和其它非法语句的任务将会被网络拒绝。此外，由于许多非法语句在代码运行之前不会被检查出来，因此ePL包含一个持续检查非法状况的运行时模块。如果发现非法状况，则会直接结束任务而不是坐等程序崩溃。

为了防止任务无限循环，传统的for，while，do loops以及goto语句都被移除了。相反，我们提供了一个repeat语句，它要求科学家们声明迭代次数的上限，并允许预处理器估计执行循环所需要的最大计算量。这样就可以精确的估计程序逻辑的"最坏执行时间"，而传统的深度嵌套，流水线式的循环则无法做到这一点。为了确保任务能及时执行，预处理器会使用"最坏执行时间"分析估算任务的计算工作量，只有当任务的"最坏执行时间"低于某特定的阈值时才会被允许提交到计算网络上，从而防止整个网络被堵塞。

此外，ePL只允许使用隔离的，受限制的且高度有限的内存空间，以防止恶意代码窃取任何其它数据或在目标平台上分配比可用资源更多的资源来攻击提供算力的平台。

6. 总结

在本文中我们讨论了 志愿者计算[6] 的一些典型范例，并确定了几个可以改进的方向。因此，我们提出了一个改进的版本，XEL，它利用区块链技术以方便科学家提供额外的奖励来激励矿工们提供计算资源。这些改进吸引了许多参与者，即使他们对正在参与的科研项目没有任何兴趣--这是传统志愿者计算的初步设想。我们也简要介绍了端到端工作流程的各个方面，从创建新的计算任务到如何验证矿工们发现的解决方案。虽然和传统的志愿者计算很相似，但是由于货币激励制度可能会吸引对平台的各种不同攻击，因此必须对工作和解决方案的处理方式进行一些重要的改变。这些改变的具体细节超出了本文的讨论范围，但将在后续更详细的版本中介绍。

7. 更多

XEL是完全开源的，欢迎有兴趣的同学一起参与创建。

源代码：https://github.com/xel-software

官网：https://xel.org/

另欢迎对勤勉的翻译者进行慷慨捐赠：

比特币钱包地址：*39sr4FzJJ49ARnGsHZRxrMoqw6NFJDZMrk*

比特币现金钱包地址：*bitcoincash:pr6g2auzkcunu08chskz5uzlh3rtlf4lyshdcgjzum*

XEL钱包地址：*XEL-4GNH-9EB7-TRMD-BG5WG* 以及公钥：*2a78f77aa6c5fde82bc179fa2d05b70a6333d93960203aa30639b7c89e63066b*

引用

[1] 伯克利开放式网络计算平台 - 百度百科. [在线; 访问于2019/06/05]; 维基百科[在线; 访问于2019/06/04]

[2] 在家搜寻外星智慧 - 维基百科. [在线; 访问于2019/06/04]

[3] 互联网梅森素数大搜索 - 维基百科. [在线; 访问于2019/06/04]

[4] 货郎担问题 - 百度百科. [在线; 访问于2019/06/10]

[5] 高度并行计算 — 维基百科. [在线; 访问于2019/06/04]

[6] 志愿者计算 — 维基百科. [在线; 访问于2019/06/04]

[7] 工作量证明 — 百度百科. [在线; 访问于2019/06/10]

[8] 学习ePL. [在线; 访问于2019/06/10]

[9] 伪随机数. [在线; 访问于2019/06/10]

[10] 哈密顿图 — 百度百科. [在线; 访问于2019/06/10]