(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 105760219 A
(43)申请公布日 2016.07.13

(21)申请号 201610063538.6

(22)申请日 2016.01.29

(71)申请人 中国人民解放军信息工程大学
地址 450052 河南省郑州市高新区科学大道62号

(72)发明人 魏强 曹琰 武泽慧 柳晓龙 麻荣宽 曾杰

(74)专利代理机构 郑州大通专利商标代理有限公司 41111
代理人 陈大通

(51)Int.Cl.
G06F 9/48(2006.01)
G06F 9/38(2006.01)

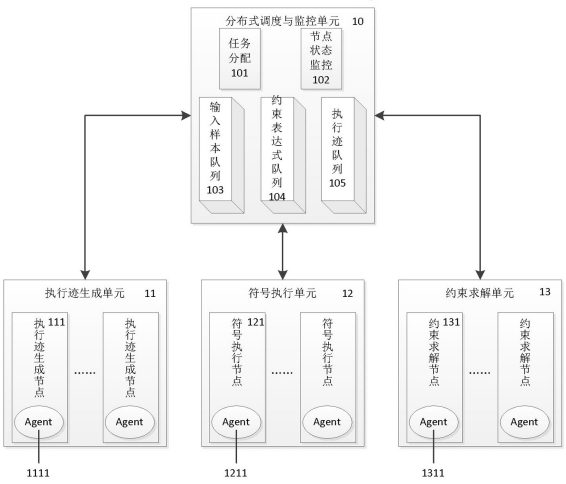
权利要求书2页 说明书6页 附图2页

(54)发明名称

基于多Agent分布式调度的并行符号执行系统

(57)摘要

本发明公开了一种基于多Agent调度的并行符号执行系统，包括：分布式调度与监控单元，用于对执行迹生成单元、符号执行单元和约束求解单元实现流水线并行调度，使得系统并行运行程度提高，负载均衡；执行迹生成单元，用于将具体输入驱动被测程序实际执行，检测是否产生异常，同时记录程序执行指令序列，通过分析转换生成中间语言表示形式的程序执行迹；符号执行单元，用于对程序执行迹进行符号化模拟执行，收集路径分支的约束条件，生成相应的约束表达式；约束求解单元，用于对约束表达式进行深度优先取反求解，生成新的输入样本，同时对取反的表达式进行标记，避免重复路径分析。本发明实现的任务调度策略可以实现流水并行，防止仅以程序执行子树为调度负载带来的负载不均衡问题。



CN 105760219 A

1.一种基于多Agent调度的并行符号执行系统,包括分布式调度与监控单元(10)、执行迹生成单元(11)、符号执行单元(12)和约束求解单元(13);其特征在于:所述的分布式调度与监控单元(10)用于任务的分配调度,监控执行迹生成单元(11)、符号执行单元(12)和约束求解单元(13)的节点空闲状态,以维持负载均衡;

所述的监控执行迹生成单元(11)包括多个执行迹生成节点(111),每个节点结构功能完全相同,可以独立执行相关功能,且都包括1个Agent(1111)用于与分布式调度与监控单元(10)通信;

所述的符号执行单元(12)包括多个符号执行节点(121),每个节点结构功能完全相同,可以独立执行相关功能,且都包括1个Agent(1211)用于与分布式调度与监控单元(10)通信;

所述的约束求解单元(13)包括多个约束求解节点(131),每个节点结构功能完全相同,可以独立执行相关功能,且都包括1个Agent(1311)用于与分布式调度与监控单元(10)通信。

2.根据权利要求1所述的基于多Agent调度的并行符号执行系统,其特征在于:所述的分布式调度与监控单元(10)包括:任务分配模块(101)、节点状态监控模块(102)、输入样本队列(103)、约束表达式队列(104)和执行迹队列(105);其中,任务分配模块(101)用于将输入样本队列(103)、约束表达式队列(104)和执行迹队列(105)中的工作任务分配到执行迹生成单元(11)、符号执行单元(12)和约束求解单元(13)等3个工作单元进行执行,并从该3个工作单元收集产生的产品放入相应的3个任务队列;节点状态监控模块(102)用于监视探测执行迹生成单元(11)、符号执行单元(12)和约束求解单元(13)中工作节点的闲/忙状态;输入样本队列(103)用于存储约束求解单元(13)生成的测试样本;约束表达式队列(104)用于存储约束符号执行单元(12)生成的约束表达式;执行迹队列(105)用于存储执行迹生成单元(11)生成的中间表示形式的程序执行指令流记录。

3.根据权利要求1所述的基于多Agent调度的并行符号执行系统,其特征在于:所述的执行迹生成节点(111)用于将具体输入驱动被测程序实际执行,检测是否产生异常,同时记录程序执行指令序列,通过分析转换生成中间语言表示形式的程序执行迹。

4.根据权利要求1所述的基于多Agent调度的并行符号执行系统,其特征在于:所述的符号执行节点(121)用于对程序执行迹进行符号化模拟执行,收集路径分支的约束条件,生成相应的约束表达式。

5.根据权利要求1所述的基于多Agent调度的并行符号执行系统,其特征在于:所述的约束求解节点(131)用于对约束表达式进行深度优先取反求解,生成新的输入样本;同时,对取反的表达式进行标记,传回分布式调度与监控单元(10)中的约束表达式队列(104)。

6.根据权利要求1所述的基于多Agent调度的并行符号执行系统,其特征在于:所述的分布式调度与监控单元(10)的具体工作方法包含以下步骤:

步骤一、节点状态监控模块(102)探测各个工作节点的闲/忙状态,如果存在执行迹生成节点(111)处于空闲,转入步骤二;如果存在符号执行节点(121)处于空闲,转入步骤三;如果存在约束求解节点(131)处于空闲,转入步骤四;如果不存在空闲状态节点,继续等待;

步骤二、任务分配模块(101)检查输入样本队列(103)是否为空,如果为空,继续等待;否则,从输入样本队列(103)取出一个输入样本传给Agent(1111);

步骤三、任务分配模块(101)检查执行迹队列(105)是否为空,如果为空,继续等待;否则,从执行迹队列(105)取出一个执行迹传给Agent(1211);

步骤四、任务分配模块(101)检查约束求解表达式队列(104)是否为空,如果为空,继续等待;否则,从约束求解表达式队列(104)取出一个约束求解表达式传给Agent(1211);

步骤五、如果输入样本队列(103)、约束表达式队列(104)和执行迹队列(105)均为空,并且执行迹生成节点(111)、符号执行节点(121)和约束求解节点(131)全部为空闲状态,则全部测试任务完成。

7.根据权利要求1所述的基于多Agent调度的并行符号执行系统,其特征在于:所述的执行迹生成单元(11)的具体方法包含以下步骤:

步骤X1、Agent(1111)接收任务分配模块(101)传来的输入样本,将自身节点状态置为“忙”;

步骤X2、启动测试任务,检测程序异常状态,记录程序指令流;

步骤X3、将记录的指令流转换成具有安全分析属性的中间语言形式,生成程序执行迹;

步骤X4、Agent(1111)将程序执行迹传入执行迹队列(105);

步骤X5、Agent(1111)将自身节点状态置为“闲”。

8.根据权利要求1所述的基于多Agent调度的并行符号执行系统,其特征在于:所述的符号执行单元(12)的具体方法包含以下步骤:

步骤Y1、Agent(1211)接收任务分配模块(101)传来的程序执行迹,将自身节点状态置为“忙”;

步骤Y2、对执行迹进行符号化模拟执行,收集路径分支约束条件,构建约束表达式;

步骤Y3、Agent(1211)将约束表达式传入约束求解表达式队列(104);

步骤Y4、Agent(1211)将自身节点状态置为“闲”。

9.根据权利要求1所述的基于多Agent调度的并行符号执行系统,其特征在于:9.所述的约束求解单元(13)的具体方法包含以下步骤:

步骤Z1、Agent(1311)接收任务分配模块(101)传来的输入样本,将自身节点状态置为“忙”;

步骤Z2、按照深度优先原则,选取约束条件进行取反,对约束求解表达式进行求解,生成新的输入样本;

步骤Z3、将取反的约束条件标记后,将新的约束表达式传入约束求解表达式队列(104);

步骤Z4、将新生成的输入样本传入输入样本队列(103);

步骤Z5、Agent(1311)将自身节点状态置为“闲”。

基于多Agent分布式调度的并行符号执行系统

技术领域

[0001] 本发明是软件脆弱性并行检测系统，属于软件工程技术领域，具体的涉及一种基于多Agent调度的并行符号执行系统。

背景技术

[0002] 并行符号执行技术的出现，是为了缓解符号执行存在的路径空间爆炸问题。符号执行有可能成为解决软件自动化测试问题的核心方法。但是，随着人们对软件功能应用的需求不断提高，单个软件的规模及其复杂性也随之持续扩张。由于软件分支数目和循环次数巨大，存在着指数级增长的执行路径，导致符号执行在实际应用中会遇到潜在的路径爆炸问题，这已成为符号执行应用的瓶颈。为了有效缓解符号执行中路径空间爆炸的问题，可以利用计算能力更高的硬件体系架构提高运算能力，比如分布式多处理器、多核、众核、云计算等，这就需要传统符号执行具有并行执行能力。

[0003] 并行符号执行技术需要依赖分布式硬件平台实现，分布式任务调度策略和任务分解方式直接影响并行效率。已有的并行符号执行方式是每个分布式节点具有相同的符号执行引擎和约束求解器，即节点间同构，完成完全相同的功能，只是将程序执行子树作为负载分配到不同的工作节点，实现并行符号执行任务。这种方式结构、通信实现简单，但是由于程序执行子树形态未知，导致负载分配很不均衡，影响并行效率。

发明内容

[0004] 本发明针对目前基于并行符号执行测试调度策略存在由于程序执行子树形态未知，导致负载分配很不均衡，影响并行效率的问题，提出一种基于多Agent调度的并行符号执行系统。

[0005] 本发明的技术方案是：一种基于多Agent调度的并行符号执行系统，包括分布式调度与监控单元(10)、执行迹生成单元(11)、符号执行单元(12)和约束求解单元(13)；所述的分布式调度与监控单元(10)用于任务的分配调度，监控执行迹生成单元(11)、符号执行单元(12)和约束求解单元(13)的节点空闲状态，以维持负载均衡。

[0006] 所述的监控执行迹生成单元(11)包括多个执行迹生成节点(111)，每个节点结构功能完全相同，可以独立执行相关功能，且都包括1个Agent(1111)用于与分布式调度与监控单元(10)通信。

[0007] 所述的符号执行单元(12)包括多个符号执行节点(121)，每个节点结构功能完全相同，可以独立执行相关功能，且都包括1个Agent(1211)用于与分布式调度与监控单元(10)通信。

[0008] 所述的约束求解单元(13)包括多个约束求解节点(131)，每个节点结构功能完全相同，可以独立执行相关功能，且都包括1个Agent(1311)用于与分布式调度与监控单元(10)通信。

[0009] 所述的基于多Agent调度的并行符号执行系统所述的分布式调度与监控单元(10)

包括:任务分配模块(101)、节点状态监控模块(102)、输入样本队列(103)、约束表达式队列(104)和执行迹队列(105);其中,任务分配模块(101)用于将输入样本队列(103)、约束表达式队列(104)和执行迹队列(105)中的工作任务分配到执行迹生成单元(11)、符号执行单元(12)和约束求解单元(13)等3个工作单元进行执行,并从该3个工作单元收集产生的产品放入相应的3个任务队列;节点状态监控模块(102)用于监视探测执行迹生成单元(11)、符号执行单元(12)和约束求解单元(13)中工作节点的闲/忙状态;输入样本队列(103)用于存储约束求解单元(13)生成的测试样本;约束表达式队列(104)用于存储约束符号执行单元(12)生成的约束表达式;执行迹队列(105)用于存储执行迹生成单元(11)生成的中间表示形式的程序执行指令流记录。

[0010] 所述的基于多Agent调度的并行符号执行系统,所述的执行迹生成节点(111)用于将具体输入驱动被测程序实际执行,检测是否产生异常,同时记录程序执行指令序列,通过分析转换生成中间语言表示形式的程序执行迹。

[0011] 所述的基于多Agent调度的并行符号执行系统,所述的符号执行节点(121)用于对程序执行迹进行符号化模拟执行,收集路径分支的约束条件,生成相应的约束表达式。

[0012] 所述的基于多Agent调度的并行符号执行系统,所述的约束求解节点(131)用于对约束表达式进行深度优先取反求解,生成新的输入样本。同时,对取反的表达式进行标记,传回分布式调度与监控单元(10)中的约束表达式队列(104)。

[0013] 所述的基于多Agent调度的并行符号执行系统,所述的分布式调度与监控单元(10)的具体工作方法包含以下步骤:

步骤一、节点状态监控模块(102)探测各个工作节点的闲/忙状态,如果存在执行迹生成节点(111)处于空闲,转入步骤二;如果存在符号执行节点(121)处于空闲,转入步骤三;如果存在约束求解节点(131)处于空闲,转入步骤四;如果不存在空闲状态节点,继续等待。

[0014] 步骤二、任务分配模块(101)检查输入样本队列(103)是否为空,如果为空,继续等待;否则,从输入样本队列(103)取出一个输入样本传给Agent(1111)。

[0015] 步骤三、任务分配模块(101)检查执行迹队列(105)是否为空,如果为空,继续等待;否则,从执行迹队列(105)取出一个执行迹传给Agent(1211)。

[0016] 步骤四、任务分配模块(101)检查约束求解表达式队列(104)是否为空,如果为空,继续等待;否则,从约束求解表达式队列(104)取出一个约束求解表达式传给Agent(1211)。

[0017] 步骤五、如果输入样本队列(103)、约束表达式队列(104)和执行迹队列(105)均为空,并且执行迹生成节点(111)、符号执行节点(121)和约束求解节点(131)全部为空闲状态,则全部测试任务完成。

[0018] 所述的基于多Agent调度的并行符号执行系统,所述的执行迹生成单元(11)的具体方法包含以下步骤:

步骤X1、Agent(1111)接收任务分配模块(101)传来的输入样本,将自身节点状态置为“忙”;

步骤X2、启动测试任务,检测程序异常状态,记录程序指令流;

步骤X3、将记录的指令流转换成具有安全分析属性的中间语言形式,生成程序执行迹;

步骤X4、Agent(1111)将程序执行迹传入执行迹队列(105);

步骤X5、Agent(1111)将自身节点状态置为“闲”。

[0019] 所述的基于多Agent调度的并行符号执行系统，所述的符号执行单元(12)的具体方法包含以下步骤：

步骤Y1、Agent(1211)接收任务分配模块(101)传来的程序执行迹，将自身节点状态置为“忙”；

步骤Y2、对执行迹进行符号化模拟执行，收集路径分支约束条件，构建约束表达式；

步骤Y3、Agent(1211)将约束表达式传入约束求解表达式队列(104)；

步骤Y4、Agent(1211)将自身节点状态置为“闲”。

[0020] 所述的基于多Agent调度的并行符号执行系统，所述的约束求解单元(13)的具体方法包含以下步骤：

步骤Z1、Agent(1311)接收任务分配模块(101)传来的输入样本，将自身节点状态置为“忙”；

步骤Z2、按照深度优先原则，选取约束条件进行取反，对约束求解表达式进行求解，生成新的输入样本；

步骤Z3、将取反的约束条件标记后，将新的约束表达式传入约束求解表达式队列(104)；

步骤Z4、将新生成的输入样本传入输入样本队列(103)；

步骤Z5、Agent(1311)将自身节点状态置为“闲”。

[0021] 本发明的有益效果是：1、本发明基于多Agent调度的并行符号执行系统，采用异构的分布式环境，实现流水式的并行调度策略，防止仅以程序执行子树为调度负载带来的负载不均衡问题，提升并行效率。

[0022] 2、本发明基于多Agent调度的并行符号执行系统，Agent技术管理灵活，特别适用于动态的、异构的分布式计算环境，成为分布式技术研究的一个重要领域，也是一个非常具有发展潜力的研究方向。

附图说明

[0023] 图1为本发明的系统结构示意图；

图2 分布式任务调度流程图。

具体实施方式

[0024] 实施例1：结合图1，一种基于多Agent调度的并行符号执行系统，包括分布式调度与监控单元10、执行迹生成单元11、符号执行单元12和约束求解单元13；所述的分布式调度与监控单元10用于任务的分配调度，监控执行迹生成单元11、符号执行单元12和约束求解单元13的节点空闲状态，以维持负载均衡；

所述的监控执行迹生成单元11包括多个执行迹生成节点111，每个节点结构功能完全相同，可以独立执行相关功能，且都包括1个Agent1111用于与分布式调度与监控单元10通信；其中，执行迹生成节点111用于将具体输入驱动被测程序实际执行，检测是否产生异常，同时记录程序执行指令序列，通过分析转换生成中间语言表示形式的程序执行迹。

[0025] 所述的符号执行单元12包括多个符号执行节点121，每个节点结构功能完全相同，可以独立执行相关功能，且都包括1个Agent1211用于与分布式调度与监控单元10通信；其

中，符号执行节点121用于对程序执行迹进行符号化模拟执行，收集路径分支的约束条件，生成相应的约束表达式。

[0026] 所述的约束求解单元13包括多个约束求解节点131，每个节点结构功能完全相同，可以独立执行相关功能，且都包括1个Agent1311用于与分布式调度与监控单元10通信。其中，约束求解节点131用于对约束表达式进行深度优先取反求解，生成新的输入样本。同时，对取反的表达式进行标记，传回分布式调度与监控单元10中的约束表达式队列104。

[0027] 所述的基于多Agent调度的并行符号执行系统所述的分布式调度与监控单元10包括：任务分配模块101、节点状态监控模块102、输入样本队列103、约束表达式队列104和执行迹队列105；其中，任务分配模块101用于将输入样本队列103、约束表达式队列104和执行迹队列105中的工作任务分配到执行迹生成单元11、符号执行单元12和约束求解单元13等3个工作单元进行执行，并从该3个工作单元收集产生的产品放入相应的3个任务队列；节点状态监控模块102用于监视探测执行迹生成单元11、符号执行单元12和约束求解单元13中工作节点的闲/忙状态；输入样本队列103用于存储约束求解单元13生成的测试样本；约束表达式队列104用于存储约束符号执行单元12生成的约束表达式；执行迹队列105用于存储执行迹生成单元11生成的中间表示形式的程序执行指令流记录。

[0028] 所述的基于多Agent调度的并行符号执行系统，所述的分布式调度与监控单元10）的具体工作方法包含以下步骤：

步骤一、节点状态监控模块102探测各个工作节点的闲/忙状态，如果存在执行迹生成节点111处于空闲，转入步骤二；如果存在符号执行节点121处于空闲，转入步骤三；如果存在约束求解节点131处于空闲，转入步骤四；如果不存在空闲状态节点，继续等待；

步骤二、任务分配模块101检查输入样本队列103是否为空，如果为空，继续等待；否则，从输入样本队列103取出一个输入样本传给Agent1111；

步骤三、任务分配模块101检查执行迹队列105是否为空，如果为空，继续等待；否则，从执行迹队列105取出一个执行迹传给Agent1211；

步骤四、任务分配模块101检查约束求解表达式队列104是否为空，如果为空，继续等待；否则，从约束求解表达式队列104取出一个约束求解表达式传给Agent1211；

步骤五、如果输入样本队列103、约束表达式队列104和执行迹队列105均为空，并且执行迹生成节点111、符号执行节点121和约束求解节点131全部为空闲状态，则全部测试任务完成。

[0029] 所述的基于多Agent调度的并行符号执行系统，所述的执行迹生成单元11的具体方法包含以下步骤：

步骤X1、Agent1111接收任务分配模块101传来的输入样本，将自身节点状态置为“忙”；

步骤X2、启动测试任务，检测程序异常状态，记录程序指令流；

步骤X3、将记录的指令流转换成具有安全分析属性的中间语言形式，生成程序执行迹；

步骤X4、Agent1111将程序执行迹传入执行迹队列105；

步骤X5、Agent1111将自身节点状态置为“闲”。

[0030] 所述的基于多Agent调度的并行符号执行系统，所述的符号执行单元12的具体方法包含以下步骤：

步骤Y1、Agent1211接收任务分配模块101传来的程序执行迹，将自身节点状态置为

“忙”；

步骤Y2、对执行迹进行符号化模拟执行，收集路径分支约束条件，构建约束表达式；

步骤Y3、Agent1211将约束表达式传入约束求解表达式队列104；

步骤Y4、Agent1211将自身节点状态置为“闲”。

[0031] 所述的基于多Agent调度的并行符号执行系统，所述的约束求解单元13的具体方法包含以下步骤：

步骤Z1、Agent1311接收任务分配模块101传来的输入样本，将自身节点状态置为“忙”；

步骤Z2、按照深度优先原则，选取约束条件进行取反，对约束求解表达式进行求解，生成新的输入样本；

步骤Z3、将取反的约束条件标记后，将新的约束表达式传入约束求解表达式队列104)；

步骤Z4、将新生成的输入样本传入输入样本队列103；

步骤Z5、Agent1311将自身节点状态置为“闲”。

[0032] 实施例2，结合图2，基于多Agent调度的并行符号执行系统包括：分布式调度与监控单元10、执行迹生成单元11、符号执行单元12和约束求解单元13。

[0033] 所述的分布式调度与监控单元10用于对执行迹生成单元、符号执行单元和约束求解单元实现流水线并行调度。进行作业调度的目的是使得系统并行运行程度提高，避免出现空闲状态的节点。并行调度服的三个任务队列的生产者和消费者情况如下：1、输入样本队列103的生产者是约束求解单元13，消费者是执行迹生成单元11；2、执行迹队列105的生产者是执行迹生成单元11，消费者是符号执行单元12；3、约束表达式队列104的生产者是符号执行单元12，消费者是约束求解单元13。这些功能分别由任务分配模块101和节点状态监控模块102完成。

[0034] 分布式任务调度具体包括如下步骤，

步骤S101：系统初始化，节点状态监控模块102分别将执行迹生成节点111、符号执行节点121和约束求解节点131置为“空闲状态”；

步骤S102：判断初始输入样本队列是否为空，如果为空，测试结束；否则，转入步骤S103；

步骤S103：任务分配模块101从输入样本队列选取样本发送给第i个空闲的执行迹生成节点，开始测试；

步骤S104：节点状态监控模块102置第i个执行迹生成节点状态为“忙”；

步骤S105：判断输入样本队列、约束表达式队列和执行迹队列是否全部为空，如果不全为空，转入步骤S106；否则，转入步骤S122；

步骤S106：分布式调度与监控单元10等待接收Agent的消息，如收到消息，转入步骤S107；否则，继续等待；

步骤S107：判断消息来源于哪类工作节点，如果来自执行迹生成节点111，转入步骤S108；如果来自符号执行节点121，转入步骤S114；如果来自约束求解节点121，转入步骤S118；

步骤S108：判断消息的任务类型，如果是请求新任务，转入步骤S109；如果是提交完成的任务，转入步骤S113；

步骤S109：判断输入样本队列103是否为空，如果为空，转入步骤S110；否则，转入步骤

S111；

步骤S110：节点状态监控模块102将请求消息源节点状态置为“空闲”，转入步骤S105；

步骤S111：任务分配模块101从输入样本队列103选取任务发送给消息源节点，开始执行测试；

步骤S112：节点状态监控模块102将消息源节点状态置为“忙”，转入步骤S105；

步骤S113：将消息源节点生成的执行迹加入执行迹队列105，转入步骤S110；

步骤S114：判断消息的任务类型，如果是请求新任务，转入步骤S109；如果是提交完成的任务，转入步骤S117；

步骤S115：判断执行迹队列105是否为空，如果为空，转入步骤S110；否则，转入步骤S116；

步骤S116：任务分配模块101从执行迹队列选取任务发送给消息源节点，开始符号执行，转入步骤S112；

步骤S117：将消息源节点生成的约束表达式加入约束表达式队列104，转入步骤S110；

步骤S118：判断消息的任务类型，如果是请求新任务，转入步骤S119；如果是提交完成的任务，转入步骤S121；

步骤S119：判断约束表达式队列104是否为空，如果为空，转入步骤S110；否则，转入步骤S120；

步骤S120：从约束表达式队列104选取任务发送给消息源节点，开始约束求解，转入步骤S112；

步骤S121：将消息源节点生成的测试样本加入输入样本队列(S121)，转入步骤S110；

步骤S122：节点状态监控模块102判断三类工作节点是否全部“空闲”，如果不全为“空闲”，转入步骤S106；否则，结束。

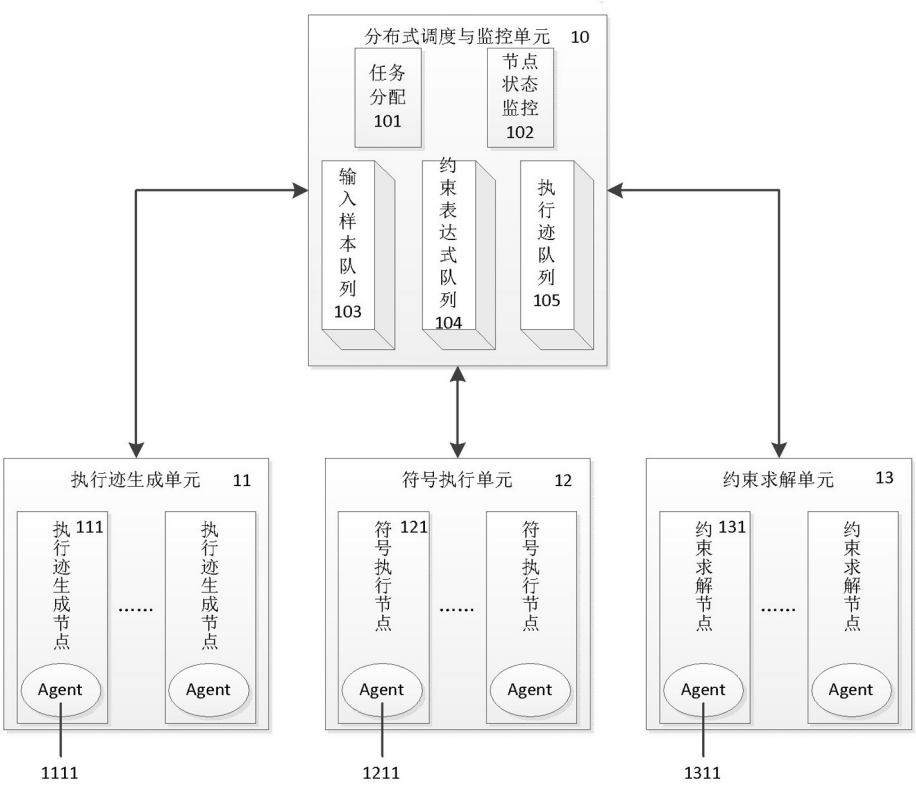

分布式调度与监控单元 10
任务分配 101
节点状态监控 102
输入样本队列 103
约束表达式队列 104
执行迹队列 105
执行迹生成单元 11
执行迹生成节点 111
执行迹生成节点
Agent
Agent
1111
符号执行单元 12
符号执行节点 121
符号执行节点
Agent
Agent
1211
约束求解单元 13
约束求解节点 131
约束求解节点
Agent
Agent
1311

图1

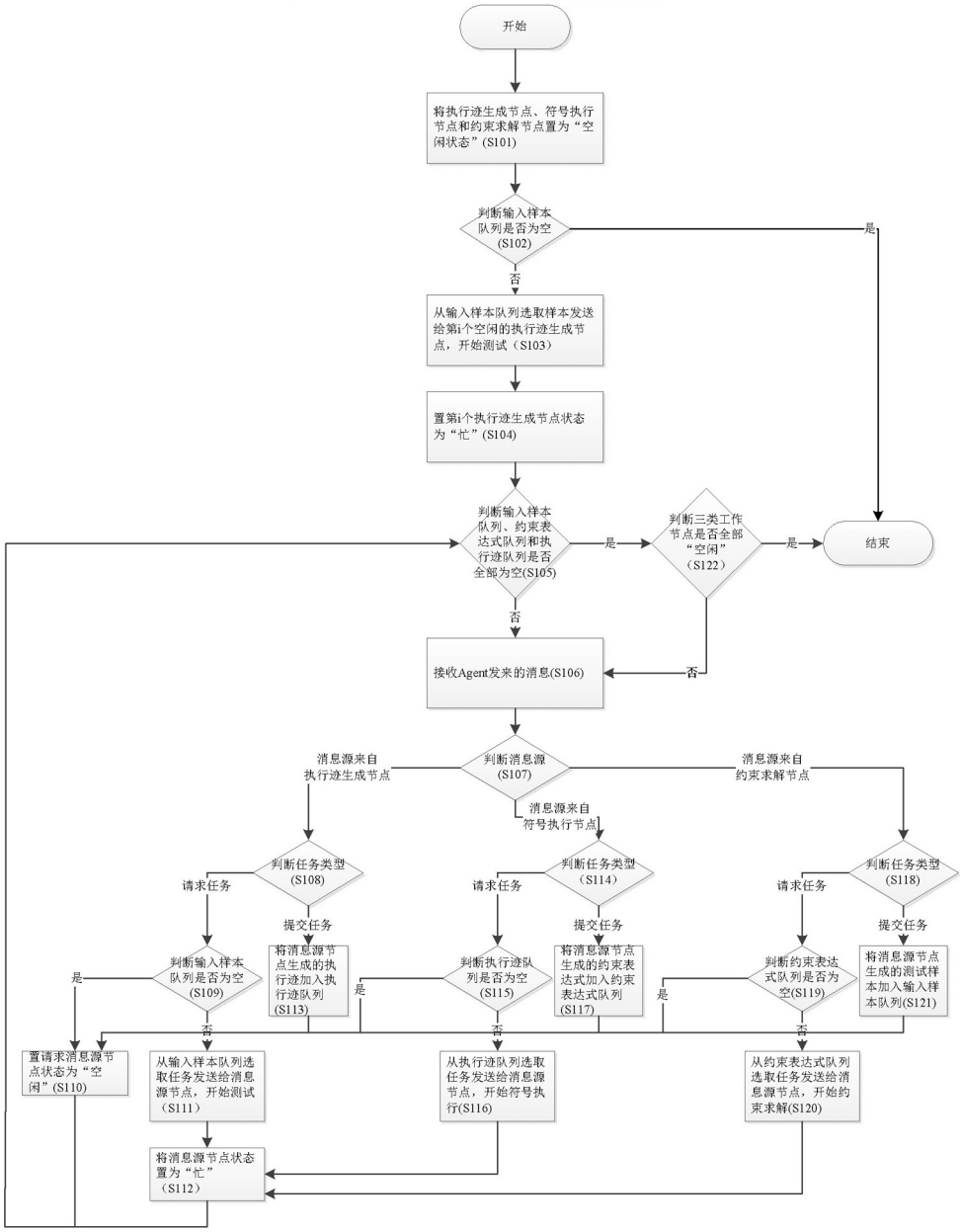

开始
将执行迹生成节点、符号执行节点和约束求解节点置为“空闲状态”(S101)
判断输入样本队列是否为空(S102)
是
否
从输入样本队列选取样本发送给第i个空闲的执行迹生成节点，开始测试（S103）
置第i个执行迹生成节点状态为“忙”(S104)
判断输入样本队列、约束表达式队列和执行迹队列是否全部为空(S105)
是
判断三类工作节点是否全部“空闲”（S122）
是
结束
否
否
接收Agent发来的消息(S106)
判断消息源(S107)
消息源来自执行迹生成节点
消息源来自符号执行节点
消息源来自约束求解节点
判断任务类型(S108)
请求任务
提交任务
判断输入样本队列是否为空(S109)
是
否
将消息源节点生成的执行迹加入执行迹队列(S113)
置请求消息源节点状态为“空闲”(S110)
从输入样本队列选取任务发送给消息源节点，开始测试（S111）
判断任务类型(S114)
请求任务
提交任务
判断执行迹队列是否为空(S115)
是
否
将消息源节点生成的约束表达式加入约束表达式队列(S117)
从执行迹队列选取任务发送给消息源节点，开始符号执行(S116)
判断任务类型(S118)
请求任务
提交任务
判断约束表达式队列是否为空(S119)
是
否
将消息源节点生成的测试样本加入输入样本队列(S121)
从约束表达式队列选取任务发送给消息源节点，开始约束求解(S120)
将消息源节点状态置为“忙”（S112）

图2