(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 102169455 A
(43) 申请公布日 2011.08.31

(21) 申请号 201010117750.9

(22) 申请日 2010.02.26

(71) 申请人 国际商业机器公司
地址 美国纽约

(72) 发明人 周韡 郭少聃 鞠琳 王建秋
任党恩

(74) 专利代理机构 北京市中咨律师事务所
11247
代理人 于静 杨晓光

(51) Int.Cl.
G06F 11/36(2006.01)

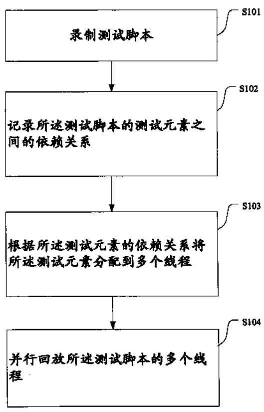
权利要求书 2 页 说明书 6 页 附图 8 页

(54) 发明名称

用于软件性能测试的调试方法和系统

(57) 摘要

本发明公开一种用于软件性能测试的调试方法和系统，该方法包括：录制测试脚本；记录所述测试脚本的测试元素之间的依赖关系；根据所述测试元素的依赖关系将所述测试元素分配到多个线程；以及并行回放所述测试脚本的多线程。本发明的方法和系统能够加速调试环节的速度，提高测试效率。



1. 一种用于软件性能测试的调试方法，包括：

录制测试脚本；

记录所述测试脚本的测试元素之间的依赖关系；

根据所述测试元素的依赖关系将所述测试元素分配到多个线程；以及

并行回放所述测试脚本的多个线程。

2. 根据权利要求 1 所述的方法，其中记录所述测试脚本的测试元素之间的依赖关系进一步包括将所述测试元素的依赖关系记录为以所述测试元素作为节点组织成的拓扑结构。

3. 根据权利要求 2 所述的方法，进一步包括从所述录制的测试脚本中获得所述测试脚本的测试元素的响应时间；以及

根据所述测试元素的依赖关系将所述测试元素分配到多个线程进一步包括利用所述测试元素的响应时间和依赖关系确定所述测试元素的关键路径，并根据所述关键路径和依赖关系将所述测试元素分配到多个线程。

4. 根据权利要求 1-3 中任一项所述的方法，并行回放所述测试脚本的多个线程进一步包括利用线程状态转换控制逻辑管理所述多个线程的状态。

5. 根据权利要求 4 所述的方法，其中利用线程状态转换控制逻辑管理所述多个线程的状态进一步包括监视所述多个线程所包含的测试元素的状态。

6. 根据权利要求 5 所述的方法，其中利用线程状态转换控制逻辑管理所述多个线程的状态进一步包括可视化地显示所述多个线程所包含的测试元素的状态。

7. 根据权利要求 5 所述的方法，利用线程状态转换控制逻辑管理所述多个线程的状态进一步包括：如果接收到测试元素处于错误状态的信息，使包含该测试元素的线程进入阻塞状态；以及如果接收到该测试元素的错误被排除的通知，使处于阻塞状态的该线程进入执行状态。

8. 根据权利要求 5 所述的方法，利用线程状态转换控制逻辑管理所述多个线程的状态进一步包括：如果接收到测试元素处于异常状态的信息，使包含该测试元素的线程进入悬挂状态；以及如果接收到测试元素的异常被排除的通知，使处于悬挂状态的该线程进入执行状态。

9. 根据权利要求 5 所述的方法，利用线程状态转换控制逻辑管理所述多个线程的状态进一步包括：如果接收到测试元素处于正常状态的通知，使包含该测试元素的线程进入执行状态。

10. 根据权利要求 5 所述的方法，利用线程状态转换控制逻辑管理所述多个线程的状态进一步包括：如果接收到线程中包含的全部测试元素执行完毕后的信息，使线程进入完成状态。

11. 一种用于软件性能测试的调试系统，包括：

录制模块，被配置为录制测试脚本；

记录模块，被配置为记录所述测试脚本的测试元素之间的依赖关系；

分配模块，被配置为根据所述测试元素的依赖关系将所述测试元素分配到多个线程；以及

回放模块，被配置为并行回放所述测试脚本的多个线程。

12. 根据权利要求 11 所述的系统，其中记录模块进一步被配置为将所述测试元素的依

赖关系记录为以所述测试元素作为节点组织成的拓扑结构。

13. 根据权利要求 12 所述的系统，还包括响应时间获取模块，被配置为从所述录制的测试脚本中获得所述测试脚本的测试元素的响应时间；以及

分配模块进一步被配置为利用所述测试元素的响应时间和依赖关系确定所述测试元素的关键路径，并根据所述关键路径和依赖关系将所述测试元素分配到多个线程。

14. 根据权利要求 11-13 中任一项所述的系统，其中回放模块进一步包括线程管理模块，被配置为利用线程状态转换控制逻辑管理所述多个线程的状态。

15. 根据权利要求 14 所述的系统，其中线程管理模块进一步包括监视模块，被配置为监视所述多个线程所包含的测试元素的状态。

16. 根据权利要求 15 所述的系统，其中线程管理模块进一步包括显示模块，被配置为可视化地显示所述多个线程所包含的测试元素的状态。

17. 根据权利要求 15 所述的系统，其中线程管理模块进一步被配置为：如果接收到测试元素处于错误状态的信息，使包含该测试元素的线程进入阻塞状态；以及如果接收到该测试元素的错误被排除的通知，使处于阻塞状态的该线程进入执行状态。

18. 根据权利要求 15 所述的系统，其中线程管理模块进一步被配置为：如果接收到测试元素处于异常状态的信息，使包含该测试元素的线程进入悬挂状态；以及如果接收到测试元素的异常被排除的通知，使处于悬挂状态的该线程进入执行状态。

19. 根据权利要求 15 所述的系统，其中线程管理模块进一步被配置为：如果接收到测试元素处于正常状态的通知，使包含该测试元素的线程进入执行状态。

20. 根据权利要求 15 所述的系统，其中线程管理模块进一步被配置为：如果接收到线程中包含的全部测试元素执行完毕后的信息，使线程进入完成状态。

用于软件性能测试的调试方法和系统

技术领域

[0001]　本发明涉及软件测试，特别涉及用于软件性能测试的调试方法和系统。

背景技术

[0002]　随着越来越多的人依托互联网从事日常业务操作，应用程序的性能测试在成功的电子商务解决方案中变得至关重要。性能测试的过程是一个负载测试和压力测试的过程，即逐渐增加负载，直到系统的瓶颈或者不能接收的性能点，通过综合分析交易执行指标和资源监控指标来确定系统并发性能的过程。负载测试 (Load Testing) 是确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统组成部分的相应输出项，例如通过量、响应时间、CPU 负载、内存使用等来决定系统的性能。压力测试 (Stress Testing) 是通过确定一个系统的瓶颈或者不能接收的性能点，来获得系统能提供的最大服务级别的测试。

[0003]　在对端到端 (end-to-end) 软件应用系统进行性能测试时，通常经由录制产生初始测试脚本，然后测试人员需要结合实际测试要求对其进行编辑完善，其中一项重要工作就是进行数据关联（或参数化），使得测试脚本在执行实际测试时更具通用性，并通过回放数据关联后的测试脚本来查看数据关联的正确性。以最普遍的浏览器 / 服务器应用为例，当录制测试脚本时，记录客户端向服务器端发出的请求和服务器端相应的响应，每个请求和响应组成一个测试元素，然后进行数据关联。很多公司开发了工具和方法来进行应用程序的性能测试。例如，HP Load Runner 的 VUGen，虽然自动测试工具可以基于某种规则识别和自动关联一部分数据，但是很多情况下仍有遗漏和错误的关联发生，测试人员需要检查并通过脚本调试结果验证来确保数据关联的完备性。当测试情景复杂或者测试者经验不足时，数据关联会经常发生错误，需要重复调试进行数据关联，每做一次数据关联都要重新回放所有测试元素，因此调试过程中，浪费了大量时间进行反复回放。

[0004]　因此，需要一种高效的软件性能测试的并行调试方法和系统。

发明内容

[0005]　基于上述问题，本发明提供一种用于软件性能测试的并行调试方法和系统。

[0006]　根据本发明的第一方面，提供一种用于软件性能测试的调试方法，该方法包括：录制测试脚本；记录所述测试脚本的测试元素之间的依赖关系；根据所述测试元素的依赖关系将所述测试元素分配到多个线程；以及并行回放所述测试脚本的多线程。

[0007]　根据本发明的第二方面，提供一种用于软件性能测试的调试系统，该系统包括：录制模块，被配置为录制测试脚本；记录模块，被配置为记录所述测试脚本的测试元素之间的依赖关系；分配模块，被配置为根据所述测试元素的依赖关系将所述测试元素分配到多个线程；以及回放模块，被配置为并行回放所述测试脚本的多个线程。

附图说明

[0008]　结合附图，通过参考下列详细的示例性实施例的描述，将会更好地理解本发明本身、优选的实施方式以及本发明的目的和优点，其中：

[0009]　图 1 示出根据本发明一个实施例的用于软件性能测试的并行调试方法；

[0010]　图 2 示出将某测试脚本实例所包含的测试元素作为节点组织成拓扑结构图；

[0011]　图 3 示出根据本发明一个实施例的根据测试元素的依赖关系将测试元素分配到多线程的方法流程图；

[0012]　图 4 示出根据本发明一个实施例的根据测试元素的响应时间和依赖关系将测试元素分配到多线程的流程图；

[0013]　图 5 示出应用图 4 的方法将图 2 所示的测试元素分配到多个线程的实例；

[0014]　图 6 示出根据本发明一个实施例的管理线程执行状态的状态转换图；

[0015]　图 7 示出可视化地显示测试元素的状态的示意图；

[0016]　以及

[0017]　图 8 示出根据本发明一个实施例的用于软件性能测试的并行调试的系统的功能框图。

具体实施方式

[0018]　以下结合附图描述根据本发明实施例的用于软件性能测试的并行调试方法和系统。

[0019]　本发明的基本思想是利用对端到端软件应用录制的测试脚本的测试元素的内在依赖关系，将测试元素重新组织编排到多个线程，测试期间，并行回放测试脚本的多个线程，调试时出现错误的线程停下来，而不影响其它线程的执行，从而加速调试环节的速度，提高测试效率。

[0020]　图 1 示出根据本发明一个实施例的用于软件性能测试的调试方法，该方法包括：在步骤 S101，录制测试脚本；在步骤 S102，记录所述测试脚本的测试元素之间的依赖关系；在步骤 S103，根据所述测试元素的依赖关系将所述测试元素分配到多个线程；以及在步骤 S104，并行回放所述测试脚本的多个线程。

[0021]　具体地，在步骤 S101，以最普遍的浏览器 / 服务器应用为例，为了建立测试脚本，测试工程师可以利用虚拟测试工具例如 HP LoadRunner 的 VUGen 来记录客户端的请求和服务器对应的响应，当录制测试脚本时，性能测试工具会拦截客户端（浏览器）与服务器端（网站服务器）之间的对话，并且通通记录在录制日志 (recording log) 中，产生测试脚本。在录制日志中可以找到浏览器与服务器之间的所有对话，包含通讯内容、日期、时间、浏览器的请求、服务器的响应内容等等。

[0022]　在步骤 S102，可以利用性能测试工具对测试脚本进行分析并记录测试脚本的测试元素之间的依赖关系，所谓测试元素之间的依赖关系，即，如果在后测试元素的请求中的某些参数的值需要从先前某个测试元素的响应中的内容获得，则这两个测试元素之间具有依赖关系。

[0023]　在步骤 S103，根据测试元素的依赖关系将测试元素分配到多个线程，为了直观反映出测试元素之间的依赖关系，可以将所述测试元素的依赖关系记录为以所述测试元素作为节点组织成的拓扑结构图，该拓扑结构图定义为 G = (N,E)，其中 N 表示测试元素的节点

集合 $\{n_0, n_1 ... n_i\}$，E 表示有向边的集合 $\{e_{i,j}\}$，$e_{i,j}$ 表示从节点 n_i 到节点 n_j 的有向边，表明节点 n_j 依赖于节点 n_i，其中节点 n_j 称之为节点 n_i 的子节点，节点 n_i 称之为节点 n_j 的父节点，图 2 示出将某测试脚本实例所包含的测试元素作为节点组织成拓扑结构图，例如 $e_{0,1}$ 表示从节点 n_0 到节点 n_1 的有向边，表明节点 n_1 依赖于节点 n_0，其中节点 n_1 称之为节点 n_1 的子节点，节点 n_0 称之为节点 n_1 的父节点，考虑将每个测试元素作为细粒度的执行任务，根据测试元素的依赖关系将测试元素分配到多个线程。

[0024] 图 3 示出根据本发明一个实施例的根据测试元素的依赖关系将测试元素分配到多个线程的方法流程图，其基本思想是尽量将具有依赖关系的测试元素分配到一个线程，在多个测试元素同时与一个测试元素存在依赖关系的情况下，将其中一个测试元素添加到其父节点的测试元素所在的线程，对其余测试元素创建新的线程。具体地，在步骤 S301，建立包含所有节点的初始化列表 entryList$\{n_0, n_1 ... n_i\}$ 并将所有未添加到线程的节点标记为未检查；在步骤 S302，在 entryList 中取出一个未检查的节点 n_i；在步骤 S303，判断节点 n_i 是否有父节点，如果判断结果为是，则前进到步骤 S304，如果判断结果为否，则在步骤 S305，创建线程 t_m，将节点 n_i 添加到线程 t_m，并标记节点 n_i 为已检查；在步骤 S304，判断 n_i 的所有父节点是否都标记为已检查，如果判断结果为否，则返回到步骤 S302，如果判断结果为是，则在步骤 S306，任取节点 n_i 的一个父节点；在步骤 S307，判断节点 n_i 的该父节点所在的线程的执行队列中该父节点之后是否存在子节点，如果判断结果为否，则在步骤 S308，将节点 n_i 添加到该父节点所在线程的执行队列中，并标记节点 n_i 已检查；如果判断结果为是，则在步骤 S309，判断节点 n_i 的父节点是否都已取出；如果判断结果为否，则返回步骤 S306；如果判断结果为是，则在步骤 S310，创建新线程 t_i，将节点 n_i 添加到线程 t_i，并标记节点 n_i 已检查；前进到步骤 S311，判断 entryList 中是否存在标记为未检查的节点，如果判断结果为是，则返回到步骤 S302，如果判断结果为否，则流程结束。根据本发明实施例的方法，可以实现将测试脚本中的多个测试元素分配到多个线程中，在调试的时候，实现多个线程并行执行调试。

[0025] 根据本发明的另一个实施例，可以利用测试元素的响应时间和依赖关系确定测试元素脚本的关键路径，并根据关键路径和依赖关系将所述测试元素分配到多个线程。具体地，可以从录制的测试脚本中获取测试脚本所包含的测试元素的响应时间，测试元素的响应时间是从测试工具发出测试元素的请求到测试工具从服务器接收到响应所经历的时间，在考虑将每个测试元素作为执行任务而分配到多个线程时，可以将每个测试元素的响应时间视为执行任务的执行时间，如图 2 所示，每个测试元素的节点中都包含有该测试元素的响应时间 t_{ni}，例如，节点 n_1 所表示的测试元素的响应时间 t_{n1} 是 343ms。

[0026] 图 4 示出本发明一个实施例的根据测试元素的响应时间和依赖关系将测试元素分配到多个线程的流程图，假设从节点 n_i 出发到各个根节点有 n 条路径 $P_1, P_2 ... P_n$(n 为正整数)，其中 P_1 包含节点 $(n_i .. n_j)$，P_2 包含节点 $(n_i .. n_k)$，P_n 包含节点 $(n_i .. n_r)$，分别计算每条路径所包含的测试元素的响应时间的总和，对于路径 P_1，计算 $T_{p1} = (t_{ni} + .. t_{nj})$，对于路径 P_2，计算 $T_{p2} = (t_{ni} + .. t_{nk})$，对于路径 P_n，计算 $T_{pn} = (t_{ni} + .. t_{nk})$，定义 $level(n_i) = Max(T_{p1}, T_{p2} ... T_{pn})$。以下描述图 4 所示的流程图，在步骤 S401，将每个节点标识为未检查，并计算每个节点的 level 值，以图 2 的节点 n_6 为例，计算 $Level(n_6)$，由图所示，从 n_6 出发到达各个根节点共有 5 条路径，分别是 P_1 包含节点 (n_6, n_7)，P_2 包含节点 (n_6, n_{12})，P_3 包含节点 (n_6, n_{16})，

P_4 包含节点 (n_6，n_{11}，n_{15}，n_{16})，P_5 包含节点 (n_6，n_{11}，n_{15}，n_{20}，n_{21}，n_{24}，n_{25})。针对上述 5 条路径，分别计算每条路径上测试元素的响应时间总和，结果可得 T_{p1} = 2548ms，T_{p2} = 2625ms，T_{p3} = 2548ms，T_{p4} = 2627ms，T_{p5} = 7391ms，因此 level(n6) = 7391；在步骤 S402，建立初始化列表 entryList，所有的节点按照 level(ni) 的值从大到小降序排列；在步骤 S403，在 entryList 中找到按照降序排列的第一个未检查的节点 n_i；在步骤 S404，创建一个新的线程 t_i，将节点 n_i 添加到 t_1 的执行队列中，并标记节点 n_i 为已检查；在步骤 S405，从 n_i 开始执行宽度优先遍历，找到下一级子节点；在步骤 S406，判断节点 n_i 是否只有一个子节点，如果判断结果为否，在步骤 S407，找到所有子节点中具有最大 level(n_k) 的子节点 n_k；如果判断结果为是，在步骤S408，判断子节点 n_k 的所有父节点是否都已经被标记为已检查，如果判断结果为是，则在步骤 S409，将子节点 n_k 添加到与节点 n_i 相同线程的执行队列中，如果判断结果为否，则前进到步骤 S412；在步骤 S410，判断节点 n_i 的其它子节点的所有父节点是否都标记为已检查，如果判断结果为是，在步骤 S411，为其它子节点 n_1，n_2...n_m 分别创建新的线程 t_1，t_2...t_m(m 为正整数)，将其它子节点 n_1，n_2...n_m 分别添加到线程 t_1，t_2...t_m 的执行队列中，并将其它子节点 n_1，n_2...n_m 标记为已检查。如果判断结果为否，则在步骤 S412，判断 entryList 中是否存在标记为未检查的节点，如果判断结果为是，则返回到步骤 S402，如果判断结果为否，则流程结束。根据本发明的实施例的方法，产生一个包含测试元素最多的线程，即测试元素脚本的关键路径，由于本发明实施例的方法能够使所开线程数目最少，从而保证线程之间通信的次数最少，提高调试的效率。与其他路径相比，关键路径中的测试元素通常更接近被测系统的核心业务操作，此外，关键路径通常包含数据关联发生错误概率较大的测试元素，在调试时测试人员可以将精力更多地集中到关键路径上。本领域技术人员应该理解可以有多种方式实现根据测试元素的依赖关系将测试元素分配到多个线程，并不限于上述的实施例，可以根据实际应用有多种变化。

[0027]　图 5 示出应用图 4 的方法将图 2 所示的测试元素分配到多个线程的实例，由图 5 所示，最终将所有测试元素分配到 9 个线程，其中线程 1 是测试元素的关键路径，在调试程序时，9 个线程并行执行，测试人员可以优先调试关键路径的测试元素，一旦关键路径调试通过，依赖于关键路径的线程就相对容易调试了。

[0028]　在步骤S104，并行回放所述测试脚本的多个线程。在调试测试脚本的过程中，由于测试元素已被预先分配到多个线程中，因此彼此没有依赖关系的各个线程可以并行执行，一旦某个线程的测试元素数据关联出现错误，则该线程停止执行，由测试人员手动检查修改错误，而其它不受该测试元素影响的线程仍然可以正常执行。这样，不必在每次回放时都将所有测试元素重新执行一遍，执行完成的线程不必重新回放，在回放过程中测试人员只要集中精力将出错的测试元素修改好以保证出错的线程执行通过即可，节省了回放的时间，提高了调试的效率。

[0029]　根据本发明的一个实施例，利用线程状态转换控制逻辑来管理多个线程的执行状态，更进一步地，线程状态转换控制逻辑可以监视多个线程所包含的测试元素的状态，例如监视测试元素的状态为正常、错误或异常。图 6 示出根据本发明一个实施例的管理线程执行状态的状态转换图，如图 6 所示，每个执行线程有五个状态，分别是：初始状态、执行状态、阻塞状态、悬挂状态以及完成状态。在不同的执行条件和运行环境下，可以由线程状态转换控制逻辑来使多个线程分别处于不同的状态。线程状态转换控制逻辑的工作过程如

下：在线程被创建之后，线程处于初始状态，如果分配给该线程的所有测试元素数据关联正确，则线程状态转换控制逻辑使线程由初始状态转变为执行状态；如果接收到测试元素处于错误状态的信息，例如数据关联错误或其它影响后续测试元素执行的错误，线程状态转换控制逻辑使包含该测试元素的线程进入阻塞状态，阻塞状态需要调试人员手动干预；如果接收到测试元素处于异常状态的信息，线程状态转换控制逻辑使包含该测试元素的线程进入悬挂状态，例如接收到测试元素发出的警告信息，该警告信息并不会影响后续测试元素的继续执行，例如线程中的某一后续测试元素依赖其他若干个前序测试元素，而这些前序测试元素并未完全执行完毕，该后续测试元素发出警告信息，悬挂状态不需要手动干预就能自动进入执行状态。处于阻塞或悬挂状态的线程，在测试元素的错误或异常被排除后，向线程状态转换控制逻辑发出通知，如果接收到线程测试元素的错误或异常被排除的通知，线程状态转换控制逻辑使阻塞或悬挂状态的线程重新进入执行状态。而处于执行状态的线程再次遇到测试元素数据关联错误或出现异常，会重新进入阻塞或悬挂状态。如果接收到线程中包含的全部测试元素执行完毕后的信息，线程状态转换控制逻辑使线程进入完成状态。

[0030] 根据本发明的一个实施例，可以可视化地显示多个线程所包含的测试元素的状态。图 7 示出可视化地显示测试元素的状态的示意图，例如，在并行回放测试脚本的多个线程的过程中，如果测试元素处于正常状态，例如数据关联正确，则测试元素显示为绿色，如图所示测试元素 n_1，n_2，n_3 数据关联正确，显示绿色；如果测试元素处于错误状态，例如数据关联错误或其它影响执行的错误，则测试元素显示为红色，如图所示测试元素 n_6 数据关联错误，显示红色；如果测试元素处于异常状态，例如发出与数据关联无关的警告信息，则后续测试元素显示为黄色，如图所示由于前序测试元素并未完全执行完毕，后续测试元素 n_9，n_{10} 等发出警告信息，显示黄色。可选地，如果光标移动到出现错误或异常的测试元素附近，会弹出有关该测试元素的错误或异常信息的对话框，例如在显示红色的测试元素节点 n_6 附近弹出如下内容的对话框：测试元素节点 n_6 数据关联错误，线程 1，3，4 和 6 被阻塞，建议修改节点 n_6，该节点的运行数据是：运行时间 25 毫秒，处于线程 1 中。通过可视化地显示测试元素的状态可使调试人员对每个测试元素的状态一目了然，并即时排查错误，提高调试效率。

[0031] 基于同一发明构思，本发明还提出一种用于软件性能测试脚本的并行调试系统，图 8 示出根据本发明一个实施例的用于软件性能测试的调试系统的功能框图，该系统包括：录制模块 801，被配置为录制测试脚本，记录模块 802，记录所述测试脚本的测试元素之间的依赖关系；分配模块 803，被配置为根据所述测试元素的依赖关系将所述测试元素分配到多个线程；以及回放模块 804，被配置并行回放所述测试脚本的多个线程。

[0032] 其中记录模块 802 进一步被配置为将所述测试元素的依赖关系记录为以所述测试元素作为节点组织成的拓扑结构。

[0033] 根据本发明的一个实施例，该系统还包括响应时间获取模块，被配置为从所述录制的测试脚本中获得所述测试脚本的测试元素的响应时间；以及分配模块 802 进一步被配置为利用所述测试元素的响应时间和依赖关系确定所述测试元素的关键路径，并根据所述关键路径和依赖关系将所述测试元素分配到多个线程。

[0034] 根据本发明的一个实施例，回放模块 804 进一步包括：线程管理模块，被配置为利

用线程状态转换控制逻辑管理所述多个线程的状态；以及监视模块，被配置为监视所述多个线程所包含的测试元素的状态。

[0035] 其中线程管理模块进一步被配置为：如果接收到测试元素处于错误状态的信息，使包含该测试元素的线程进入阻塞状态；如果接收到该测试元素的错误被排除的通知，使处于阻塞状态的该线程进入执行状态；如果接收到测试元素处于异常状态的信息，使包含该测试元素的线程进入悬挂状态；如果接收到测试元素的异常被排除的通知，使处于悬挂状态的该线程进入执行状态；如果接收到测试元素处于正常状态的通知，使包含该测试元素的线程进入执行状态；以及如果接收到线程中包含的全部测试元素执行完毕后的信息，使线程进入完成状态。

[0036] 根据本发明的一个实施例，线程管理模块进一步包括显示模块，被配置为可视化地显示所述多个线程所包含的测试元素的状态。

[0037] 应当理解，本发明的至少某些方面可以可替代地以程序产品实现。定义有关本发明的功能的程序可以通过各种信号承载介质被传送到数据存储系统或计算机系统，所述信号承载介质包括但不限于，不可写存储介质（例如，CD-ROM）、可写存储介质（例如，软盘、硬盘驱动器、读/写 CD ROM、光介质）以及诸如包括以太网的计算机和电话网络之类的通信介质。因此应当理解，在此类信号承载介质中，当携带或编码有管理本发明中的方法功能的计算机可读指令时，代表本发明的可替代实施例。本发明可以硬件、软件、固件或其组合的方式实现。本发明可以集中的方式在一个计算机系统中实现，或以分布方式实现，在这种分布方式中，不同的部件分布在若干互连的计算机系统中。适于执行本文中描述的方法的任何计算机系统或其它装置都是合适的。优选地，本发明以计算机软件和通用计算机硬件的组合的方式实现，在这种实现方式中，当该计算机程序被加载和执行时，控制该计算机系统而使其执行本发明的方法，或构成本发明的系统。

[0038] 上面出于举例说明的目的，给出了本发明的优选实施例的说明。优选实施例的上述说明不是穷尽的，也不打算把本发明局限于公开的明确形式，显然鉴于上述教导，许多修改和变化是可能的。对本领域的技术人员来说显而易见的这种修改和变化包括在由附加的权利要求限定的本发明的范围内。

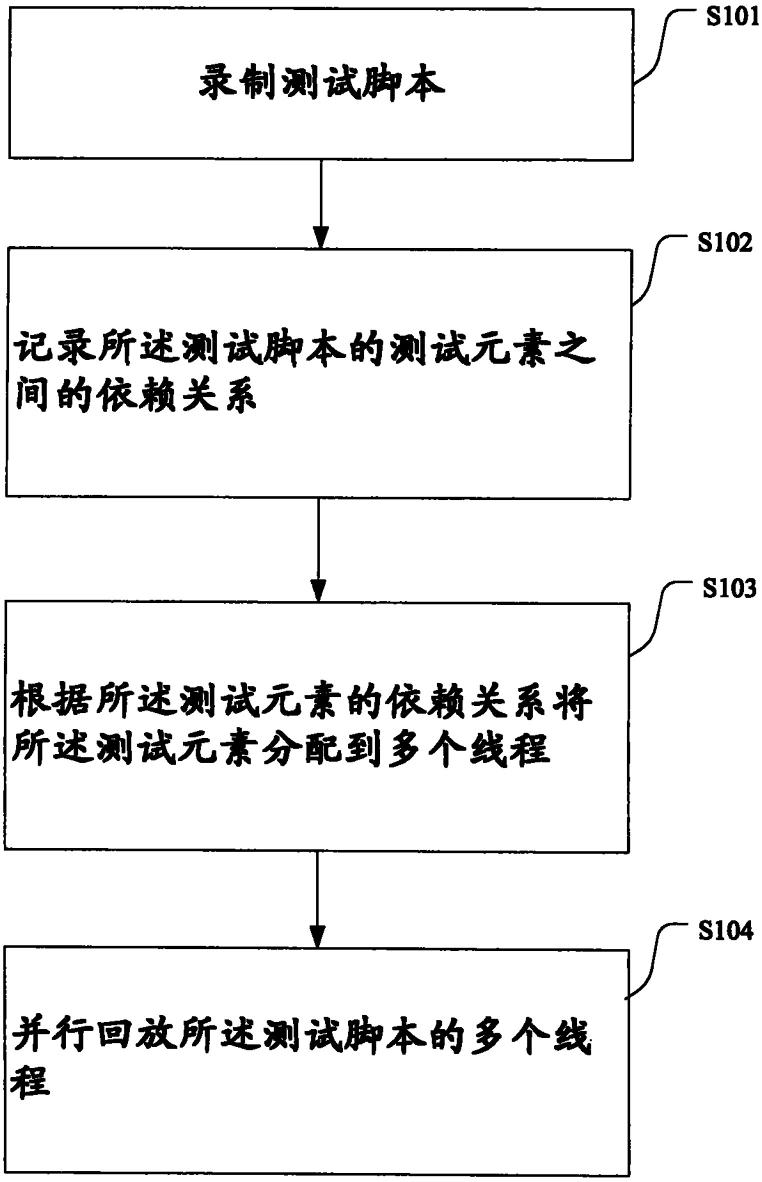

S101
录制测试脚本
S102
记录所述测试脚本的测试元素之间的依赖关系
S103
根据所述测试元素的依赖关系将所述测试元素分配到多个线程
S104
并行回放所述测试脚本的多个线程

图 1

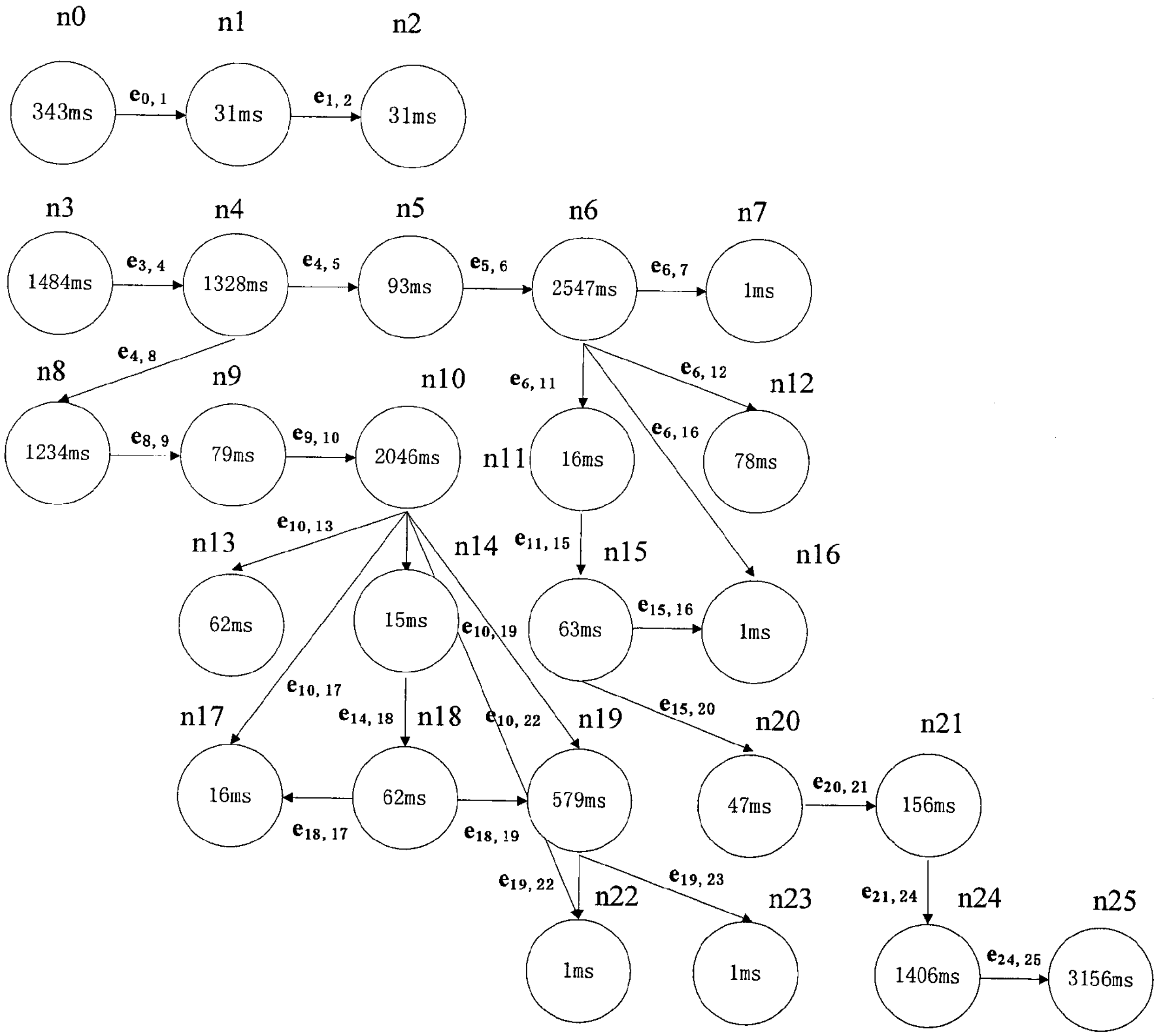

n0
343ms
e0, 1
n1
31ms
e1, 2
n2
31ms
n3
1484ms
e3, 4
n4
1328ms
e4, 5
n5
93ms
e5, 6
n6
2547ms
e6, 7
n7
1ms
e4, 8
n8
1234ms
e8, 9
n9
79ms
e9, 10
n10
2046ms
e6, 11
e6, 12
e6, 16
n11
16ms
n12
78ms
e10, 13
n13
62ms
n14
15ms
e11, 15
n15
63ms
n16
1ms
e15, 16
e10, 19
e10, 17
e14, 18
n17
16ms
n18
62ms
e10, 22
n19
579ms
e15, 20
n20
47ms
e20, 21
n21
156ms
e18, 17
e18, 19
e19, 22
e19, 23
n22
1ms
n23
1ms
e21, 24
n24
1406ms
e24, 25
n25
3156ms

图 2

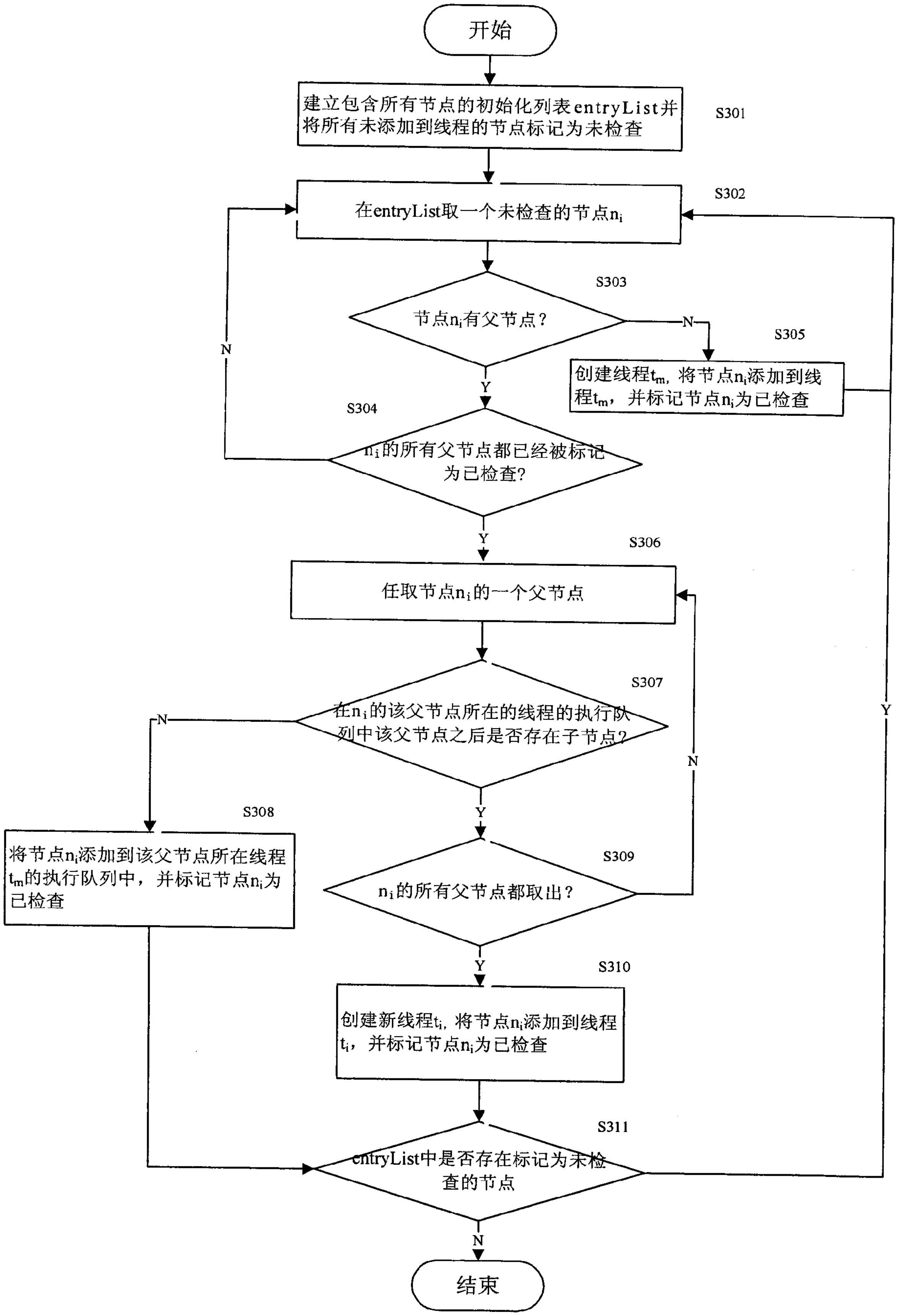

开始
建立包含所有节点的初始化列表entryList并将所有未添加到线程的节点标记为未检查
S301
在entryList取一个未检查的节点n_i
S302
节点n_i有父节点？
S303
N
S305
创建线程t_m，将节点n_i添加到线程t_m，并标记节点n_i为已检查
Y
S304
n_i的所有父节点都已经被标记为已检查?
N
Y
S306
任取节点n_i的一个父节点
S307
在n_i的该父节点所在的线程的执行队列中该父节点之后是否存在子节点?
N
Y
S308
将节点n_i添加到该父节点所在线程t_m的执行队列中，并标记节点n_i为已检查
S309
n_i的所有父节点都取出？
N
Y
S310
创建新线程t_i，将节点n_i添加到线程t_i，并标记节点n_i为已检查
S311
entryList中是否存在标记为未检查的节点
Y
N
结束

图 3

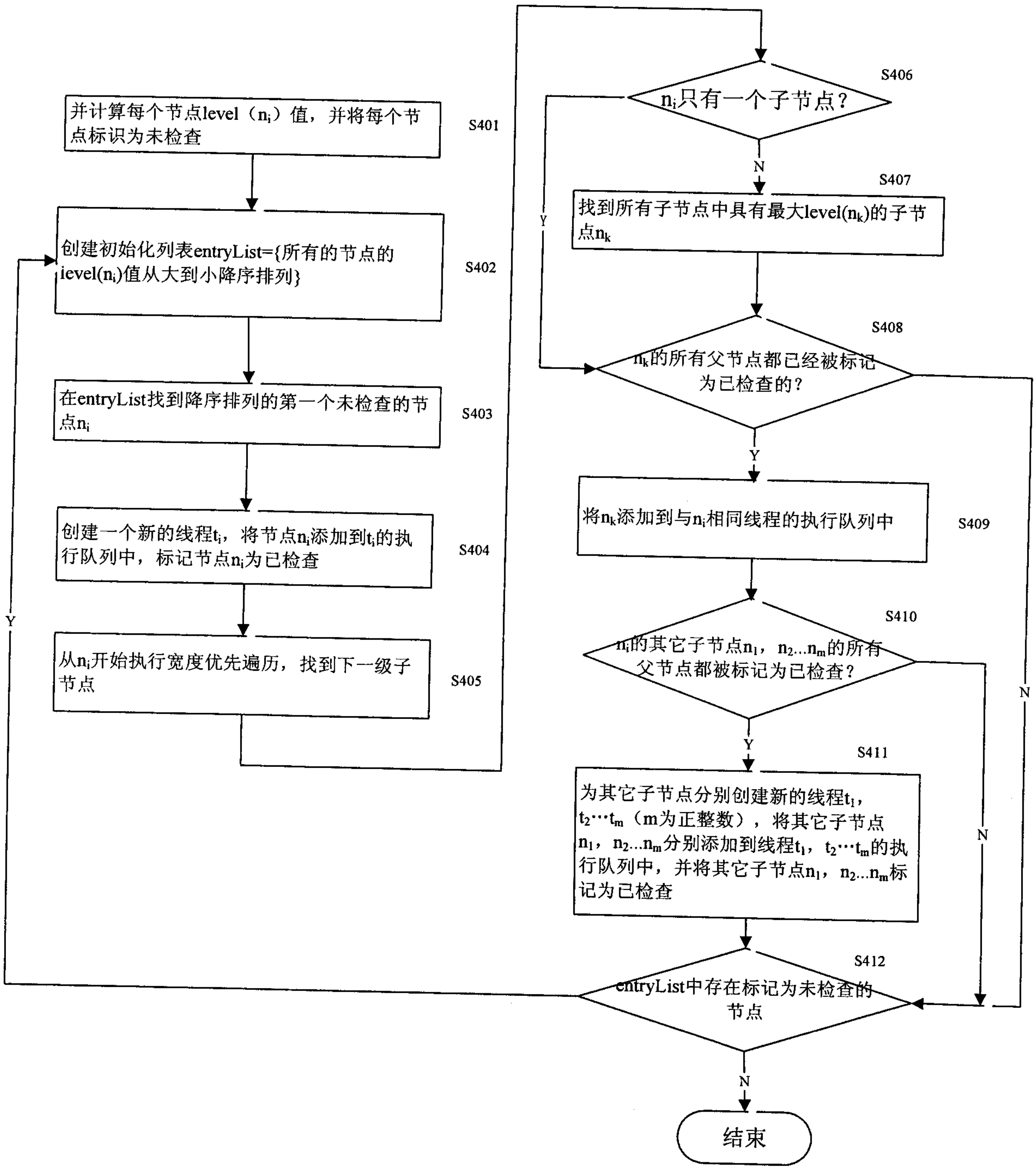

并计算每个节点level（n_i）值，并将每个节点标识为未检查
S401
创建初始化列表entryList={所有的节点的ievel(n_i)值从大到小降序排列}
S402
在entryList找到降序排列的第一个未检查的节点n_i
S403
创建一个新的线程t_i，将节点n_i添加到t_i的执行队列中，标记节点n_i为已检查
S404
从n_i开始执行宽度优先遍历，找到下一级子节点
S405
n_i只有一个子节点？
S406
N
Y
找到所有子节点中具有最大level(n_k)的子节点n_k
S407
n_k的所有父节点都已经被标记为已检查的？
S408
Y
N
将n_k添加到与n_i相同线程的执行队列中
S409
n_i的其它子节点n_1，n_2...n_m的所有父节点都被标记为已检查？
S410
Y
N
为其它子节点分别创建新的线程t_1，t_2…t_m（m为正整数），将其它子节点n_1，n_2...n_m分别添加到线程t_1，t_2…t_m的执行队列中，并将其它子节点n_1，n_2...n_m标记为已检查
S411
entryList中存在标记为未检查的节点
S412
Y
N
结束

图 4

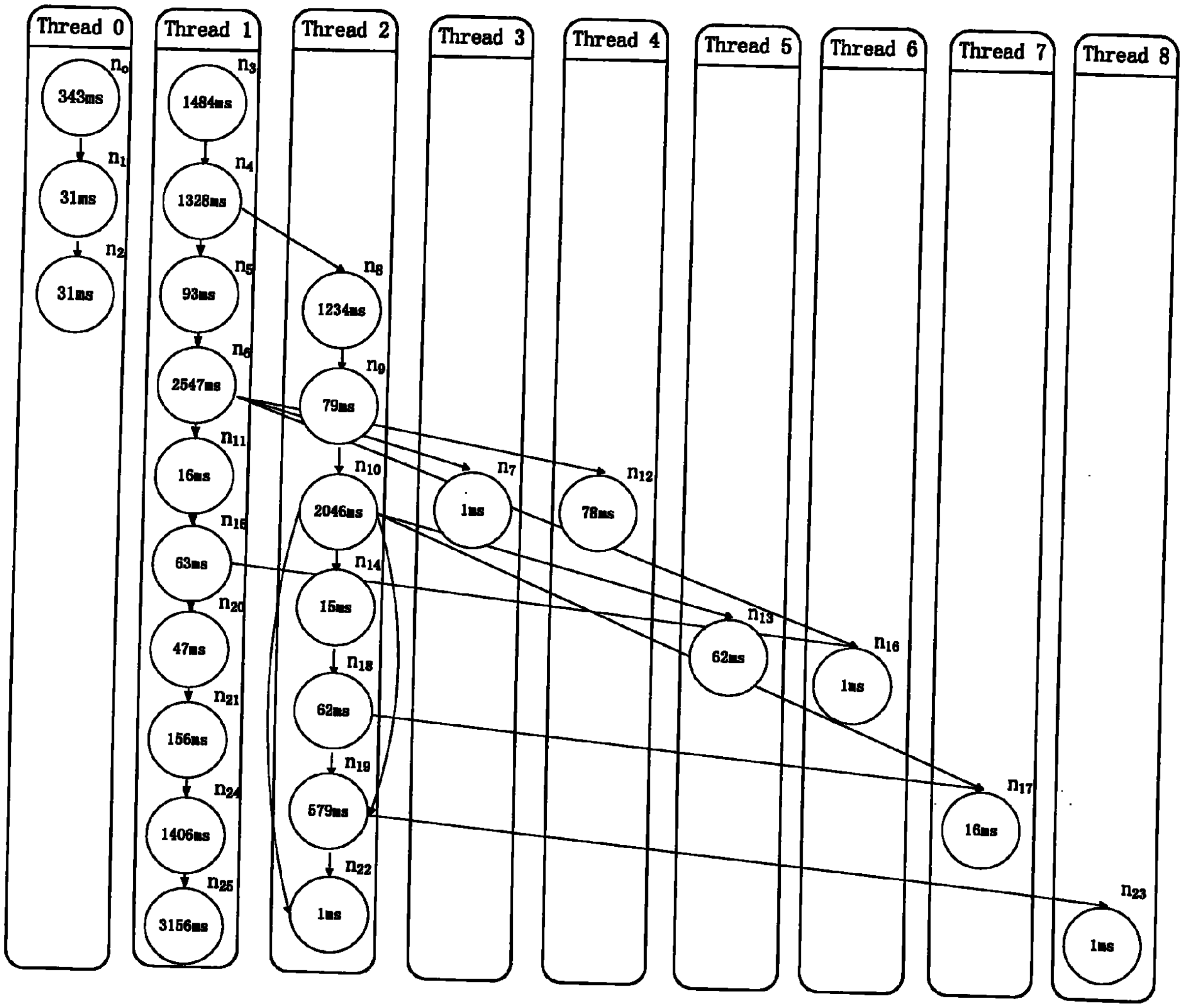

Thread 0
Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Thread 6
Thread 7
Thread 8
343ms
31ms
31ms
1484ms
1328ms
93ms
2547ms
16ms
63ms
47ms
156ms
1406ms
3156ms
1234ms
79ms
2046ms
15ms
62ms
579ms
1ms
1ms
78ms
62ms
1ms
16ms
1ms

图 5

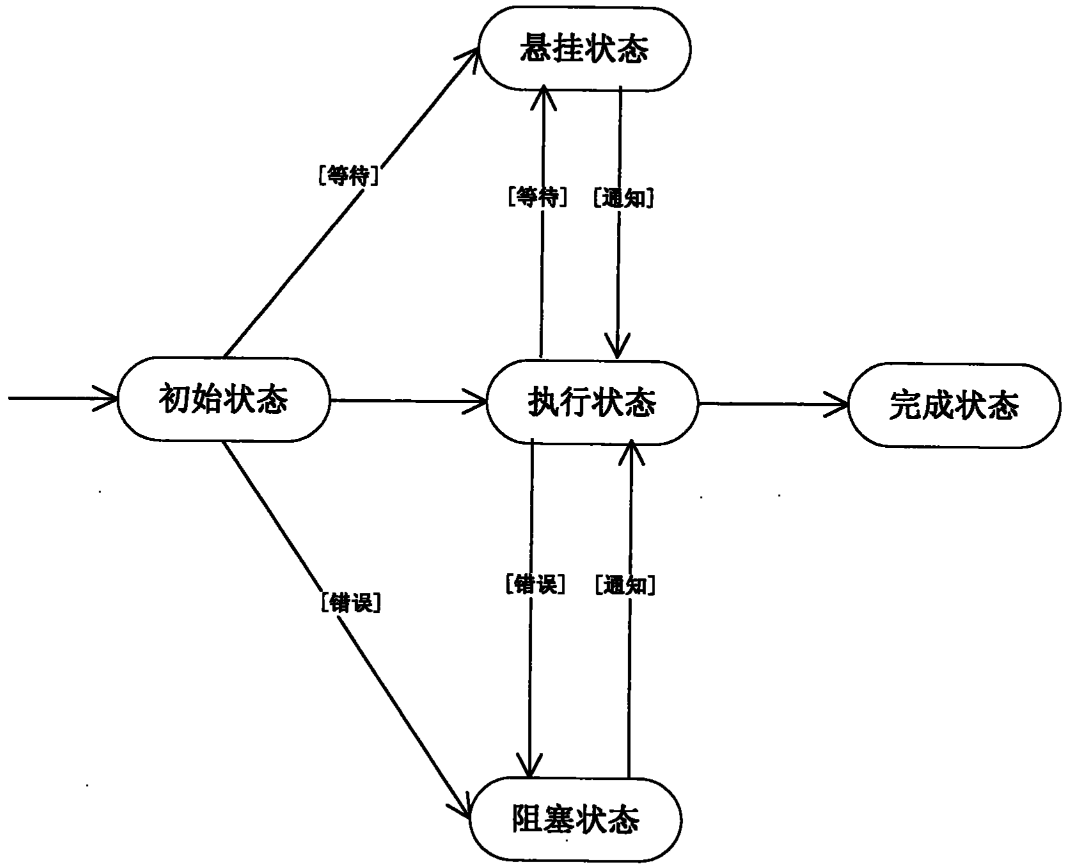


图 6

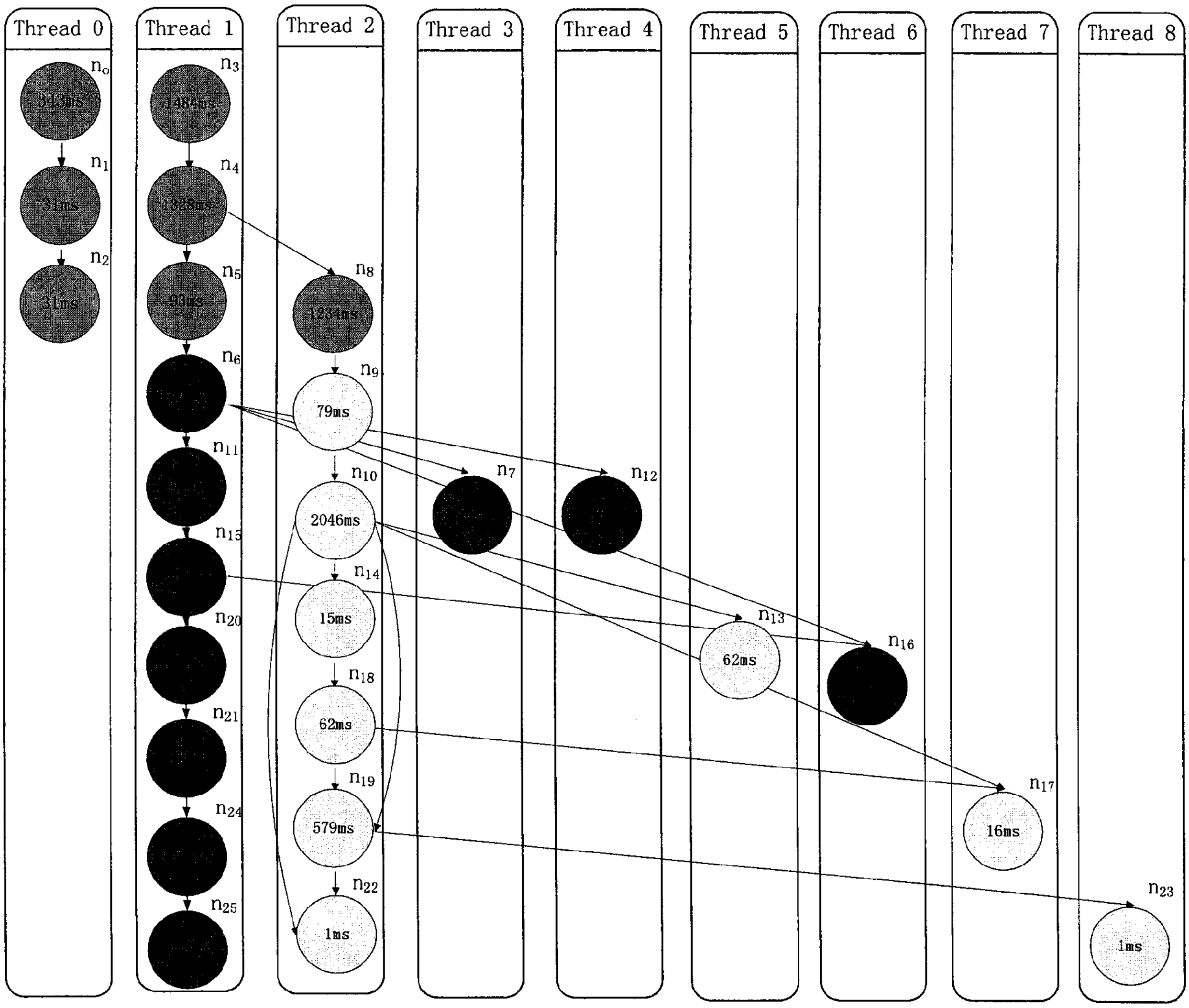

Thread 0
Thread 1
Thread 2
Thread 3
Thread 4
Thread 5
Thread 6
Thread 7
Thread 8
n0
343ms
n1
31ms
n2
31ms
n3
1484ms
n4
1328ms
n5
93ms
n6
n11
n15
n20
n21
n24
n25
n8
1234ms
n9
79ms
n10
2046ms
n14
15ms
n18
62ms
n19
579ms
n22
1ms
n7
n12
n13
62ms
n16
n17
16ms
n23
1ms

图 7

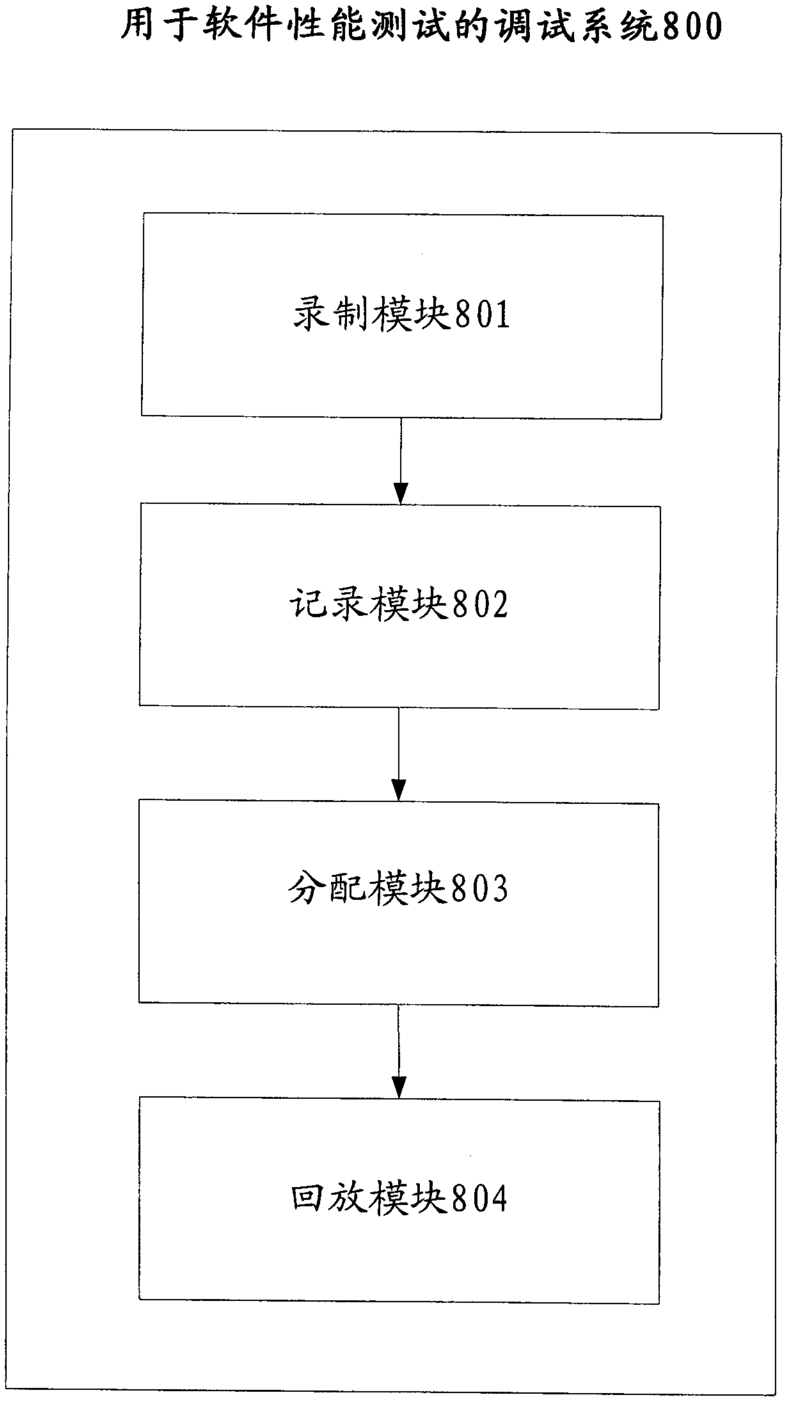


图 8