



[12] 发明专利说明书

专利号 ZL 200510107568.4

[45] 授权公告日 2008 年 10 月 22 日

[11] 授权公告号 CN 100428189C

[22] 申请日 2005.9.29

US5323489A 1994.6.21

[21] 申请号 200510107568.4

WO00/45317 2000.8.3

[30] 优先权

审查员 魏 峰

[32] 2004.9.30 [33] US [31] 10/954,574

[74] 专利代理机构 中国国际贸易促进委员会专利
商标事务所

[73] 专利权人 国际商业机器公司

代理人 李镇江

地址 美国纽约

[72] 发明人 约翰·戴维斯·帕尔莫

桑迪普·马德哈夫·乌塔姆昌达尼

卡拉德哈尔·沃鲁甘迪

[56] 参考文献

CN1445675A 2003.10.1

US5555415A 1996.9.10

US5860105A 1999.1.12

CN1475911A 2004.2.18

CN1439961A 2003.9.3

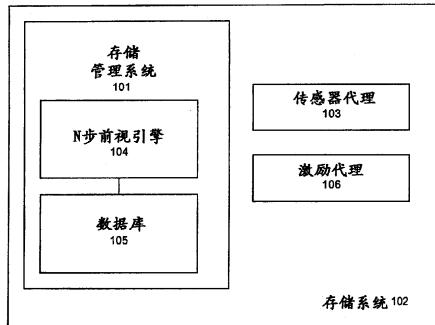
权利要求书 3 页 说明书 14 页 附图 7 页

[54] 发明名称

在基于策略的系统管理中以 N 步前视来推理
的系统

[57] 摘要

当检测到与存储系统性能目标有关的报警状态时，存储管理系统调用 N 步前视引擎以便仿真当存在有所述存储系统可以采取以便消除所述报警状态的多个动作时所述存储系统的操作。N 步前视引擎基于存储系统的当前状态产生 N 个可能的系统状态。所述的 N 个可能的状态基于所述多个动作中的每一个的成本模型。每个成本模型基于动作，所述动作的行为涵义，所述动作的资源涵义和所述动作的瞬时成本。选择产生这样的系统状态的动作，所述系统状态优化稳定性，先决条件和调用所选择的动作的瞬时成本。



1. 一种管理存储系统的方法，包括：

检测与存储系统的性能目标有关的报警状态；

调用 N 步前视引擎以便仿真当存在有所述存储系统可以采取以便消除所述的报警状态的多个动作时所述存储系统的操作；

基于所述存储系统的当前状态产生 N 个可能的系统状态；和

响应于所述报警状态被检测到，确定引起所述报警状态的所述存储系统的当前状态是否与所述存储系统的早期状态类似；如果确定所述存储系统的当前状态与所述存储系统的早期状态类似，则从存储器选择在所述早期状态下所采用的动作，否则则根据动作的成本模型及推理引擎分析对所述 N 个可能的系统状态进行分析，以此对动作进行选择。

2. 如权利要求 1 的方法，还包括调用所选择的动作。

3. 如权利要求 1 的方法，还包括存储所述的 N 个可能的系统状态。

4. 如权利要求 1 的方法，其中产生 N 个可能的状态，是基于所述的多个动作的每一个的成本模型产生每个可能的状态。

5. 如权利要求 4 的方法，其中每个成本模型基于动作，即基于所述动作的行为涵义，所述动作的资源涵义和所述动作的瞬时成本。

6. 如权利要求 1 的方法，其中产生 N 个可能的状态，是根据基于规则的系统产生每个可能的状态。

7. 如权利要求 1 的方法，其中产生 N 个可能的状态基于至少一个规范产生每个可能的状态。

8. 如权利要求 1 的方法，其中 N 是用户可选择的。

9. 如权利要求 1 的方法，还包括：

当所述存储系统的当前状态与所述 N 步前视引擎被调用时所述存储系统的状态的差异大于一个预定的差异时，终止所述的 N 步前视引擎；和

基于所述存储系统的当前状态重新调用所述的 N 步前视引擎以便

仿真操作。

10. 一种管理存储系统的系统，包括：

检测与存储系统的性能目标有关的报警状态的传感器；和
存储管理系统，其调用 N 步前视引擎以便仿真当存在有所述存储
系统可以采取以便消除所述的报警状态的多个动作时所述存储系统的
操作，所述 N 步前视引擎基于所述存储系统的当前状态产生 N 个可能
的系统状态，响应于所述传感器检测到所述报警状态，所述存储管理
系统确定引起所述报警状态的所述存储系统的当前状态是否与所述存
储系统的早期状态类似；如果确定所述存储系统的当前状态与所述存
储系统的早期状态类似，则从存储器选择在所述早期状态下所采用
的动作，否则则根据动作的成本模型及推理引擎分析对所述 N 个可能
的系统状态进行分析，以此对动作进行选择。

11. 如权利要求 10 的系统，其中所述存储管理系统调用所选择的
动作。

12. 如权利要求 10 的系统，还包括存储所述的 N 个可能的系统状
态的数据库。

13. 如权利要求 10 的系统，其中所述 N 步前视引擎基于所述存储
系统的当前状态产生 N 个可能的系统状态，是基于所述的多个动作的
每一个的成本模型产生每个可能的状态。

14. 如权利要求 13 的系统，其中每个成本模型基于动作，即基于
所述动作的行为涵义，所述动作的资源涵义和所述动作的瞬时成本。

15. 如权利要求 10 的系统，其中所述 N 步前视引擎基于所述存储
系统的当前状态产生 N 个可能的系统状态，是根据基于规则的系统产
生每个可能的状态。

16. 如权利要求 10 的系统，其中所述 N 步前视引擎基于至少一个
规范产生每个可能的状态。

17. 如权利要求 10 的系统，其中 N 是用户可选择的。

18. 如权利要求 10 的系统，其中当所述存储系统的当前状态与所
述 N 步前视引擎被调用时所述存储系统的状态的差异大于一个预定的

差异时，所述存储管理系统终止所述的 N 步前视引擎，并且基于所述存储系统的当前状态重新调用所述的 N 步前视引擎以便仿真操作。

在基于策略的系统管理中 以 N 步前视来推理的系统

技术领域

本发明涉及存储系统。更具体地，本发明涉及用于管理存储系统的系统和方法。

背景技术

基于策略的管理被视为一种允许存储管理员管理比当前管理的存储更大量的存储的万灵药（panacea）。对于基于策略的管理方法，系统管理员指定关于性能，可靠性，安全，备份，恢复等的高层策略（目标），并且存储管理软件使用规则引擎自动地将所述高层策略转换为低层的存储动作。因此如所实现的，存储管理的职责从系统管理员转移到了存储管理软件提供商。存储管理软件提供商必须处理由于大量的异质设备、商务规则、用户和存储管理动作的存在伴随而来的复杂性。软件提供商还必须确保他们的管理软件足够通用以便处理工作负荷和系统配置的改变，即，不是“脆弱的”。

在复杂的存储环境中，当特定的高层目标不被满足时，潜在地存在多种可以被采取以便纠正问题的存储管理动作。对于给定的系统状态，潜在地可以有多种适用的规则。当前，基于规则的系统通过随机地选择一种可用的规则以自组织（ad-hoc）的方式处理这种情况。对于推理引擎来说，对选择哪个特定的存储管理动作进行先验优先级划分是不容易的，这是因为每一存储管理动作具有可能潜在地使系统进入例如颠簸（thrashing）模式或进入次最优状态的复杂的副作用。此外，各个相应的动作具有必须被考虑在内的与其调用相关联的不同的成本。对这种情况的一个近似的比拟是国际象棋比赛，其中特定走步的潜在的结果直到许多步之后才会清楚。因此，国际象棋选手会尝试

在内心中提前分析许多步而不实际走任何步。

N 步前视算法被用于人工智能 (AI) 游戏理论领域，诸如国际象棋和跳棋。N 步前视算法还被用于磁盘 I/O 调度和 CPU 指令调度领域。类似地，监视实际系统以便创建 what-if 分析当前被用于数据库领域。与 N 步前视算法相关的研究可以划分为三类：(1) N 步前视的应用领域；(2) 监视信息以便创建系统模型；和 (3) 人工智能 (AI) 和规划中的 N 步前视。

在应用领域研究的类中，N 步前视实现具有用于建模和推理系统内的动作的特定于领域的语意。例如，最小化 I/O 磁盘访问次数以便优化并行 I/O 系统中的预取和高速缓存的传统的缓冲区管理算法在多个 I/O 被同时进行的并行 I/O 系统中实质上是次优的。参见例如，M.Kallahalla 等人的“Optimal prefetching and caching for parallel I/O system,” In ACM Symposium on Parallel Architectures and Algorithms, pp.219-228, 2001, 其公开了一种采用全局 L 块前视的在线算法，全局 L 块前视为该缓冲区管理算法给出了由 L 个不同请求组成的前视。

在 S. J. Beaty, “Lookahead scheduling,” ACM SIGMICRO Volume 23, Issue 1-2 (December 1992) 中，公开了一种用于执行指令的前视调度技术，即与数据相关的 DAG (DDD)。此外，根据 Beaty，结合有其它调度技术的前视可以增加产生正确的调度的可能性。

分布式系统中的仿真依赖于对模型固有的并发性的检测，其与对各个逻辑仿真过程的未来行为的预测有关。参见例如 J.W.Janneck, “Generalizing lookahead-behavior prediction in distributed simulation,” In Preceeding of twelfth workshop on Parallel and distributed simulation, pp. 12-19, 1998, 其公开了一个对使用前视算法进行行为预测的传统方法的概述。

对于磁盘中的前视调度，参见例如，T.Yeh 等人的“Competitive analysis of on-line disk scheduling,” Theory of Computing Systems, Vol.31, pp. 491-506,1998, 其分析了在线磁盘调度的问题，其中可以执

行对下面将被读的 K 个变量的前视，并且根据这种知识可以选择以何种顺序从磁盘上读所述的变量，以便最小化寻道启动时间。

在监视信息以便创建系统模型的类中，M.Selter 等人的“**Self-Monitoring and Self-Adapting Operating Systems,**”In **Proceedings of Sixth Workshop on Hot Topics in Operating Systems, May 1997** 公开了 Vino，即一种自我监视并自适应的操作系统，其对操作系统的进行连续监视，以便构造性能统计数据库。数据被适当地分类，并且执行离线分析以便构造对系统在常规行为下的特征，并且检测反常行为。组件的调整全部是预定的固定的实现。

在数据库领域，诸如 SMART 和 AutoAdmin 的方法使用查询优化器，所述的查询优化器可以为数据库中的自动索引选择进行关于“what-if”情景的推理。对于 SMART 实现，参见例如 G.M.Lohman 等人的“**DB2 advisor: An optimizer Smart Enough to Recommend Its Own Indexes,**” Proceedings, 16th IEEE conference on Data Engineering, San Diego, CA, 2000。对于 AutoAdmin，参见例如，S.Chaudhuri 等人的“**AutoAdmin “What-if” Index Analysis Utility,**” Proceedings ACM SIGMOD Conference, Seattle 1998, pp. 367-378。SMART 系统被连续地监视，并且记录每个查询的执行。优化器使用过去的历史以便创建用于所述查询中的操作的成本模型，然后所述成本模型被用于改进执行计划。

在 AI 和规划理论的类中，前视的概念被广泛地用于游戏理论，游戏理论是策略的科学，并且试图数学地和逻辑地确定“选手”将采取的动作，以便在大量“游戏”中为它们自己保证最佳的结果。研究的游戏的范围从国际象棋到抚养小孩，从网球到收购。但是所述游戏全部共享相互依赖的共同特性。即，每个参与方的结果取决于所有参与方的选择（策略）。Deep Blue，例如，被建造为每秒钟检查 2 亿个可能的走步，并且前视多致 14 轮（考虑对手走步的所有排列）。前视算法结合有搜索算法，诸如 A*，仿真退火（Simulated Annealing）法，爬山法（Hill-climbing）和前向剪枝（Forward pruning）。参见，例如，

P.Norvig, *Paradigms of AI Programming: Case Studies in Common Lisp*, 1991, 以及 H.-D.Bocker 等人的“*Interactive Problem Solving Using Log.*”

一般地在机器人技术，人工智能和控制理论的环境中提到规划理论。在机器人技术中，焦点是设计通过处理复杂的几何模型产生有用的动作的算法。在人工智能中，焦点是设计使用决策 - 理论模型计算适当的动作的系统。在控制理论中，呈现的焦点是数值地计算可行的轨迹或甚至是最佳的反馈控制规律的算法。存在有多种用于这些领域中的每一个中的前视的方法。在 E.J, Sandewall 的“*A Planning Problem Solver Based on Look-Ahead in Stochastic Game Trees*,”*Journal of the ACM (JACM)*, Volume 16, Issue 3, July 1969 中描述了将游戏理论中的前视算法映射为规划算法中的搜索启发的关注的技术。

因此，需要一种用于管理存储系统的技术，其响应报警状态，选择这样的动作，以便优化存储系统的稳定性，存储系统的先决条件和调用所选动作的瞬时成本。

发明内容

本发明提供了一种用于管理存储系统的技术，其响应报警状态，选择这样的动作，该动作优化存储系统的稳定性，存储系统的先决条件和调用所选动作的瞬时成本。

通过一种管理存储系统的方法提供了本发明的优点，其中与存储系统的性能目标有关的报警状态被检测。 N 步前视引擎被调用以便在有多个动作可以被存储系统采取以便消除所述的报警状态时仿真存储系统的操作。 N 个可能的系统状态被基于存储系统的当前状态产生。基于所述多个动作中的每一个的成本模型产生 N 个可能状态中的每一个。每个成本模型基于一个动作，所述动作的行为涵义(*implication*)，所述动作的资源涵义以及所述操作的瞬时成本。选择这样的动作，其产生优化存储系统的稳定性，所述存储系统的先决条件以及调用所述

被选择的动作的瞬时成本的可能的系统状态。然后调用所选择的动作。当存储系统的当前状态与所述 N 步前视引擎被调用时存储系统的状态之间的差异大于一个预定的差异时，所述的 N 步前视引擎被终止，并且基于所述存储系统的当前状态，所述的 N 步前视引擎被重新调用以便仿真操作。

在本发明的一个示例实施例中，被存储的 N 个可能的系统状态可以被存储。因此，当所述报警状态被检测到时，引起所述报警状态的存储系统当前的状态是否与存储系统的早期状态类似被确定。如果是的，从存储器中选择一个动作，被选择的动作是当存储系统的状态与存储系统当前状态类似时先前被选择的。

本发明还提供了一种用于管理存储系统的系统。所述系统包括传感器，存储管理系统和存储所述 N 个可能的系统状态的数据库。所述传感器检测与存储系统的性能目标有关的报警状态。存储管理系统调用 N 步前视引擎以便模拟当有多个动作可以被存储系统采用以便消除所述的报警状态时存储系统的操作。N 步前视引擎基于存储系统的当前状态产生 N 个可能的系统状态。所述 N 步前视引擎基于所述多个动作中的每一个动作的成本模型产生各个可能的状态。每个成本模型基于一个动作，所述动作的行为涵义，所述动作的资源涵义和所述操作的瞬时成本。所述存储管理系统选择这样的动作，该动作产生优化存储系统的稳定性，存储系统的先决条件以及调用所选择的动作的瞬时成本的可能的系统状态。所述存储管理系统还调用被选择的动作。当存储系统的当前状态与所述 N 步前视引擎被调用时存储系统的状态之间的差异大于一个预定的差异时，所述存储管理系统终止所述的 N 步前视引擎，并基于存储系统的当前状态重新调用 N 步前视引擎以便仿真操作。

当存储管理系统确定当所述报警状态被检测到时，引起所述报警状态的存储系统的当前状态与存储系统的早期状态类似时，存储管理系统选择一个被存储在数据库内的以前被在存储系统的状态与存储系统的当前状态类似时选择过的动作。

附图说明

在附图中以示例的方式而不是作为限制示出了本发明，其中同样的标号指示着同样的元件，其中：

图 1 给出了根据本发明的存储管理系统和存储系统的功能方框图；

图 2 给出了由吞吐量，可靠性，响应时间，安全性和可用性组成的示例的系统行为；

图 3 表示由本发明的学习模块记录的用于量化预取规范的属性的示例参数；

图 4 示出了根据本发明的 N 步前视模块的功能框；

图 5 示出了根据本发明的成本模型规范和由推理引擎得出的信息之间的交互；

图 6 示出了 n 维行为空间内的表示为向量的行为和资源涵义函数；和

图 7 示出了根据本发明由 N 步前视模块得出的 O (x*K) 个可能的系统状态的回溯树。

具体实施方式

传统的存储管理系统访问系统的当前状态，并且然后采取动作，以便向更有利的状态前进。相反，本发明提供了用于存储管理的前视范例，即基于当前的系统状态采取使系统进入新的假想状态的操作。所述假想动作可以被以重复的方式施加的次数可以作为系统参数被控制。本发明的方法，此处被称为 N 步前视方法，通过仿真所有的相关系统的可观察量跟踪假想的系统状态。最后，在 N 个动作被仿真之后，本发明基于所述的 N 步前视仿真选择最佳的可用动作，并且对系统的当前实际状态施加所选择的动作。

本发明的 N 步前视机制完全与推理引擎集成在一起。可用于执行的规则集还包括由于转换到假想系统状态而可使用的规则。当系统被

仿真时，底层物理系统的行为被建模。因此，由本发明使用的系统模型本质上不是静态的。即，所述模型随着被学习引擎更新而进化。不过N步前视预测的方法可以由于外部因素的变化诸如新用户和新设备的加入，系统组件的故障，在进行N步前视仿真的过程中指定目标的改变导致系统进入次最优状态。因此当确定实际系统的当前状态已经变得与以前假设的所述仿真的开始状态相去甚远时，仿真被终止并且另一个仿真被开始。

图1给出了根据本发明的存储管理系统101和存储系统102的功能方框图。存储管理系统101典型地不在存储系统102的直接数据路径内。存储管理系统101通过传感器代理103监视存储系统102。存储管理系统101分析并规划当感测到存储系统性能问题时所采取的纠正动作。存储系统102或是直接地或是通过存储管理系统101间接地经由传感器代理103监视各种系统资源。当被监视的资源超过预先定义的阈值时传感器代理103发出警报。所述警报被发送到存储管理系统101。

响应警报，存储管理系统101确定是否存在可能会纠正引起所述警报的问题的多个竞争动作。当存在多个动作时，存储管理系统101调用仿真引擎，N步前视引擎104，其基于接收自存储管理系统101的输入对存储器系统102的操作建模。由N步前视引擎104产生的输出被存储在仿真输出数据库105内。最初，N步前视引擎104进行检查以便查看当前状态是否与过去已经被处理的其它状态类似，并且如果有类似的状态，过去采取的纠正动作被从数据库105中检索出来，并且将以前采取的纠正动作输出到存储管理系统101。当存储器系统102的当前状态是新的状态时，N步前视引擎104开始仿真。N步前视引擎104执行由用户规定的深度为“N”的仿真。越深的仿真深度提供对潜在的纠正操作的副作用的更好的分析。激励代理106被用于采取所选择的纠正动作。

本发明的技术适用于基于规则的策略管理和宣告型方法两者。本发明的动作可以分为三部分：相关动作的成本模型的分析，N步前视

引擎分析和推理引擎分析。动作的成本模型是其行为涵义和其资源涵义的组合物。为了简化成本模型，本发明将成本模型处理为单独的函数。每个系统都具有可以改变其行为的动作。此处使用的术语“行为”表示系统可观察到的特性。可以使用抽象诸如服务质量（Qos）目标，事务属性等规定所述的特性。例如，图 2 给出了由吞吐量，可靠性，响应时间，安全性和可用性构成的示例的系统行为 200。此处使用的术语“系统状态”表示系统的细节，即资源的利用，系统事件和工作负荷特性。资源的利用被以使用的 cpu, I/O 和网络带宽表示。系统事件可以规定系统状态诸如磁盘 95% 已满，或错误诸如网络故障或磁盘故障。工作负荷特性包括读写比，顺序/随机等。

动作的影响是当前状态和当前行为的函数，即，将存储器资源从 256MB 增加到 512MB 与将同样的存储器资源从 512MB 增加到 1GB 具有不同的影响。类似地，将延迟从 8msec 改变为 4msec 需要与将延迟从 4msec 改变为 2msec 不同的系统影响。

关于动作的行为涵义 b 定义了动作对系统行为的影响。具体地，行为涵义 b 被定义为 $b(\text{Current state}, \text{Current Behavior}, \% \text{invocation value}) \rightarrow \% \text{Change in Behavior}$ 。行为涵义 b 是取决于参数诸如当前行为，当前系统状态和所述动作的调用程度的复合函数。系统的当前行为包括这样的参数，诸如吞吐量，延迟，可用性和安全性。类似地，所述系统状态是资源和应用访问模式的快照（snapshot），每一个都是多个变量的集合。一般地，一个动作与这些变量的一个小的子集有关。所述子集的一些变量可能是主要的或预期的影响，而所述子集的其它变量可能是副作用。例如，预取一般具有对于吞吐量的主要影响，对于延迟的副作用，并且可能不会以任意方式影响安全性。

动作的资源涵义 r 被定义为 $r(\text{Current state}, \% \text{invocation value}) \rightarrow \text{New System state}$ 。动作对系统状态的影响主要被在资源方面量化，即，存在有对观察到的工作负荷特性的二次影响，但是为了简单起见所述影响可以被忽略。资源涵义 r 与行为涵义 b 相比直观得多。

除了涵义函数之外，每个动作具有一个相关联的变量，以便表示

调用所述动作的瞬时开销。可以通过监视系统并且测量各个操作在各个系统资源上的开销得出瞬时成本 C 的值。为了简单起见，本发明将动作的开销分为一类，并且分配数值 1, 10 和 100 以便分别表示低，中等和高开销。因此，瞬时开销变量是粗略的估计值并且将所述动作分为一类而不是精确地测量所述的开销，精确地测量所述的开销将增加复杂性，这是因为一些动作的开销是被涉及的数据的量的函数。瞬时成本值被通过一个规范指定。

有两种得出成本模型函数的可行的方法。在基于规则的系统的情况下，成本模型函数被仅通过学习得出。虽然搜索空间可以是大的，并且因此所需的迭代的数目可以是高的，但是其仍然是一种得出成本模型函数的可行的方法。得出成本模型函数的另一种可能性是通过使用用于定义动作的宣告性规范。在第二种方法中，规范形成了用于所述学习处理的蓝图，并且有助于监视和记录与特定动作相关的参数。

考虑预取动作的例子。在基于规则的系统中，存在有调用预取动作的规则。下面给出了用于调用这种预取动作的三个示例的规则。每次一个动作被调用时，系统参数就被记录并且被加到用于内插（interpolation）的学习算法的数据集。

Event: Latency_not_met

```
If {(Memory_available > 70 && FC_interconnect_available > 60)
  && (access_pattern < 0.4 sequential && read/write > 0.4)}
  Prefetch = 1.2*Prefetch
```

Event: Latency_not_met

```
If {(15 < Memory_available > 70 && FC_interconnect_available > 60)
  && (access_pattern > 0.7 sequential && read/write > 0.4)}
  Prefetch = 1.4*Prefetch
```

Event: Latency_not_met

```
If {(Memory_available > 70 && FC_interconnect_available > 60)
  && (0.4 < access_pattern < 0.7 sequential && read/write > 0.4)}
  Prefetch = 1.3*Prefetch
```

宣告性规范使用规范作为成本模型的蓝图。然后通过在所述动作每次被调用时添加信息，所述的成本模型被连续地改进。例如，所述的规范模型将动作的属性分为两组：元级属性和基础级属性。

元级属性被推理引擎使用以便在若干竞争动作之间进行选择。落在元级之下的属性包括行为涵义和前提。行为涵义列举出动作对不同可观察量的影响。在所述规范中，管理员不是定量地提供这种信息，而是使用描述性术语，诸如上，下和无。例如，`<implication dimension=throughput impact=up>`。前提描述所述动作对资源和工作负荷特性的相关性。例如，`<precond dimension=memeory, value=*>`。

基础级属性包括调用一个动作所使用的函数和参数的细节。例如，`<function name=change_prefetch_size>`。所述的函数和参数可以被使用现有的标准诸如 SMI - S 表示。

出于 N 步前视的目的，仅有元级属性是相关的。用于预取的示例的规范包括：

```
<action name = PREFETCH>

<behavior_implications>

<implication dimension = throughput impact = up >

</behavior_implications>

<preconditions>

<precond dimension = sequential/random ratio value = high >

<precond dimension = read/write ratio value = high >

<precond dimension = memory value = * >

<precond dimension = fc_bandwidth value = * >

</preconditions>

<Overhead function = low>
```

每当操作被在规则中调用时，所述的信息被记录在所述规则的上下文中。

学习引擎可以利用传统类型的机器学习算法改进所述的成本模型。在人工智能领域，学习算法被视为黑箱，其在给出前面的 n 个数据点的采样的情况下，内插第 $(n+1)$ 个数据点的信息。在基于规则的系统的情况下，学习以基于情况的推理（CBR）为基础，其中为每个动作调用记录“系统快照”。可替换地，学习可以涉及在所述规范中向元级属性添加信息。对于涵义，对可观察到的目标调用动作的影响被量化，诸如增加 20% 的预取改善了 8% 的吞吐量。对于前提，用于调用所述动作的阈值被记忆，诸如以小于 20% 的可用存储器调用预取对性能有负面影响。同样，被用作调用值中的百分比变化的函数的资

源百分比可以被记忆。

建立学习函数中的一个非一般性任务是定义所述学习函数依靠的参数。例如，在预取的情况下，吞吐量涵义是预取大小的值，可观察量（即，吞吐量）的当前值，资源状态（例如可用的存储器）的当前值和工作负荷特性（例如，顺序/随机比）的改变的函数。

图 3 表示由本发明的学习引擎记录的用于量化用于预取规范的属性的示例参数。从所述规范中得出用于所述学习函数的参数，即，被监视的资源，以及为给定动作测量的工作负荷特性。特别地，本发明使用基于情况的推理作为用于涵义和前提的学习算法。对于基础调用，本发明采用使用神经网络的增强式学习。

图 4 给出了根据本发明的 N 步前视模块 104 的功能框。N 步前视模块 104 包括成本函数分析器 401，学习引擎 402，行为和系统状态开发器 403 以及推理引擎 404。N 步前视模块 104 的输入包括由推理引擎 404 的第一迭代产生的 k 个候选动作的一个列表。此后，所述 N 步前视模块迭代地为所述 k 个候选动作中的每一个产生可能的系统状态；N 步前视模块 104 的输出是 $O(k*N)$ 个可能的系统状态。对于每个状态，记录回溯路径，所述路径的总成本和到达每个状态所需的迭代次数。

在接收到所述的 k 个候选动作之后，成本函数分析器 401 为各个动作得出所述的成本函数。每个成本函数是一个三元组 $\langle b, r, \text{开销变量} \rangle$ ，它们具有作为当前行为和系统状态的函数的值。在基于规则的系统的情况下，通过仅使用学习引擎 402 得出成本函数，即，没有用于动作模型的规范。可替换地，可以通过使用规范和由学习引擎 402 收集的数据的组合得出成本函数。例如，图 5 中示出了所述规范和学习组件之间的交互，其给出了根据本发明的成本模型规范和由推理引擎 404 得出的信息之间的交互。学习引擎 402 通过内插用于所述属性的值补充所述的规范。接着，行为和系统状态开发器 403 应用各个相应候选动作的成本模型函数，以便作为向量添加操作得出行为和系统状态的新值。图 6 给出了 n 维行为空间内的表示为向量的行为和资源涵

义函数。

作为例子，数据复制规则的行为涵义是沿着吞吐量，延迟和可用性维度的向量。该向量被表示为： $B(\text{数据复制}) = [(0.3) \text{ 吞吐量} - (0.1) \text{ 延迟} + (0.2) \text{ 可用性}]$ ，其中调用复制分别提高了 30% 的吞吐量和 20% 的可用性，并且降低了 10% 的延迟。

在每次迭代的结尾，有 $O(x^k)$ 个可能的系统状态，其中 x 是迭代深度并且 k 是候选动作的初始集的基数。对于每次迭代，新的状态（资源和行为）+ 指定的目标被输入到推理引擎 404，以便输出下一组候选动作。很可能推理引擎 404 不施加任何候选动作。该状态被称为“终点”状态并且当状态中的下列为真时发生：

所有被指定的目标被满足，且没有资源阈值事件。

注意 N 步前视模块 104 是可并行的，即，可以由 K 个代理并行地计算各个路径，所述的各个代理的每一个相应于所述的候选动作。每个代理的结果被组合并且被输入到推理引擎 404。

在 N 次迭代的结尾， $O(K^N)$ 个系统状态以及用于每个路径的回溯路径，每个路径的总成本和到达该状态所需的迭代的次数被输入到推理引擎 404。推理引擎 404 使用优化函数确定回溯树中满足先决条件并且优化稳定性和瞬时成本函数的“最佳”路径。图 7 给出了由 N 步前视模块 104 得出的 $O(x^k)$ 个可能的系统状态的回溯树 700。每个系统状态被由一个圆圈表示。空的圆圈表示中间的系统状态。黑的或被填充的圆圈表示终点状态。图 7 中还给出了示例的瞬时成本 C 。

通过优化稳定性函数，在行为和资源涵义方面产生了最“稳定”的系统状态。此处“稳定”被定义为目标和当前状态（被表示为 n 维向量空间内的向量）之间的点积。通过优化瞬时成本函数，为一个操作的调用获得最低的瞬时成本。通过优化先决条件，诸如动作的链式调用（反映为若干迭代）和同一动作的重复调用被避免了。

通过对路径的所有边进行简单的加法为每个路径计算瞬时成本函数 C 。然后同一动作的连续调用被检查，并且当被检测到时，RepeatFlag 被设置为 True。所述路径内被调用的迭代/动作的次数 I

被计算。状态（每个路径的状态）和所述目标的点积被计算。所述点积是向量之间的夹角的余弦 S。所述路径被以 S 的值的降序排序。被排序的列表的最高的 x% 的路径被保留，并且具有 RepeatFlag = True 的路径被滤除，即，被从列表删除。供最后挑选的路径被以 C 的升序重新排序。所述列表的第一个元件以及其路径的回溯被选择。N 步前视函数的结果是所选择的路径的起始的候选动作。

虽然前面已经出于清楚地理解的目的以某种详细程度说明了本发明，显然可以实现所附权利要求的范围内的某些改变和修改。因此，本实施例被视为是示例性的而不是限制性的，并且本发明不限于此处给出的细节，而是可以在所附权利要求的范围和等同物内修改。

图 1

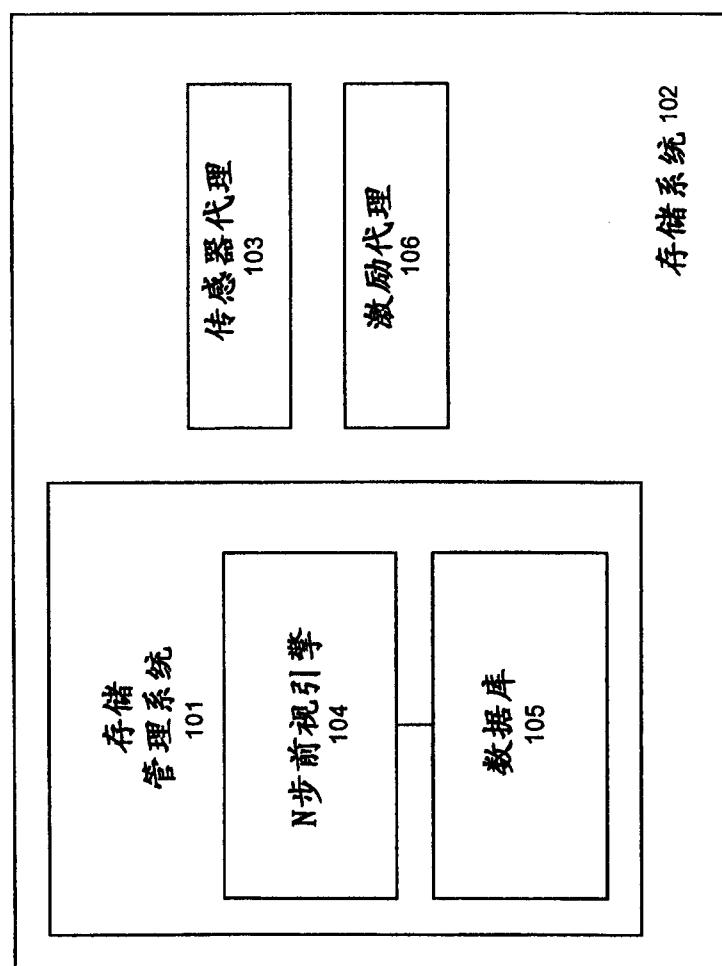


图 2

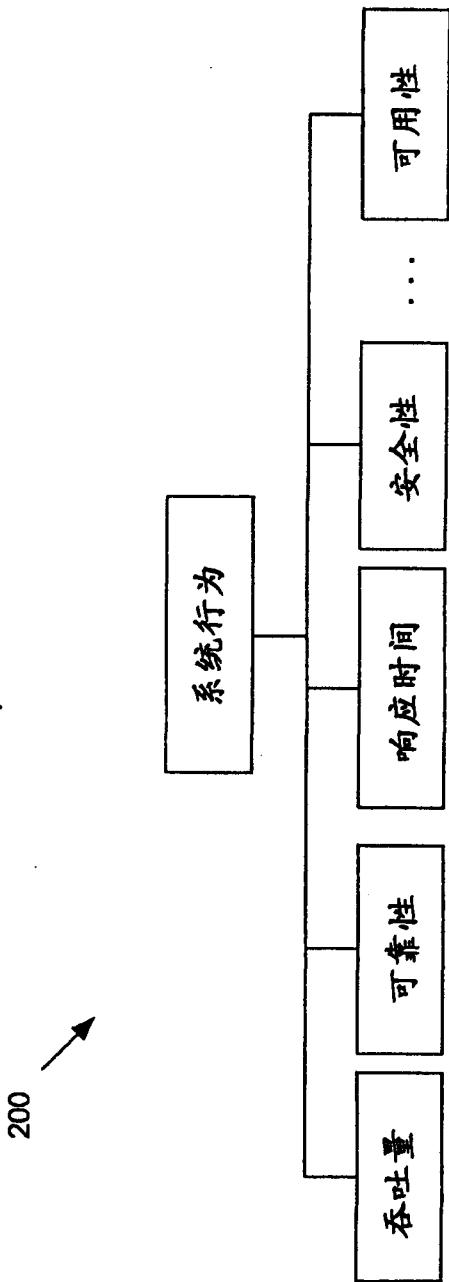


图 3

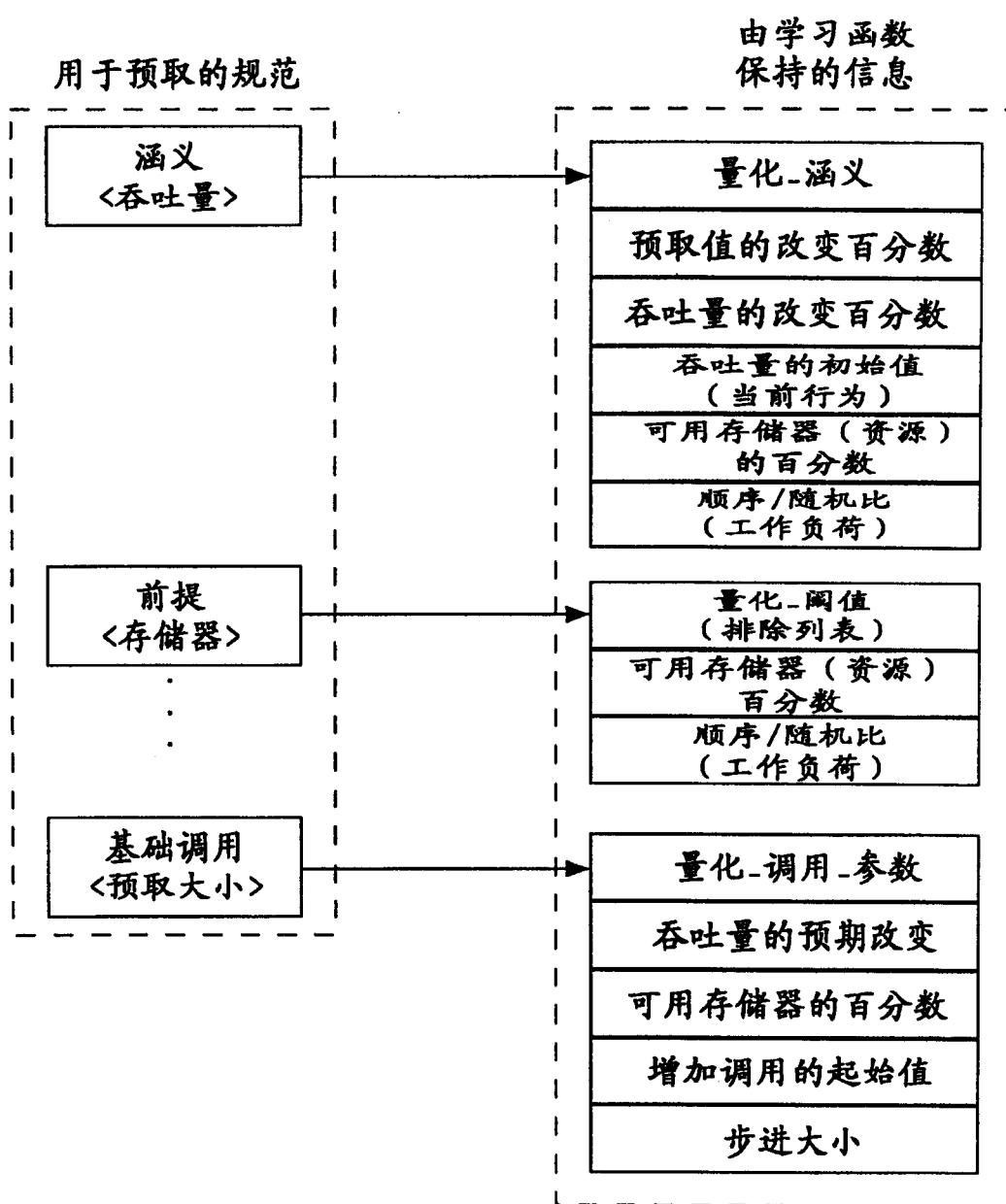


图 4

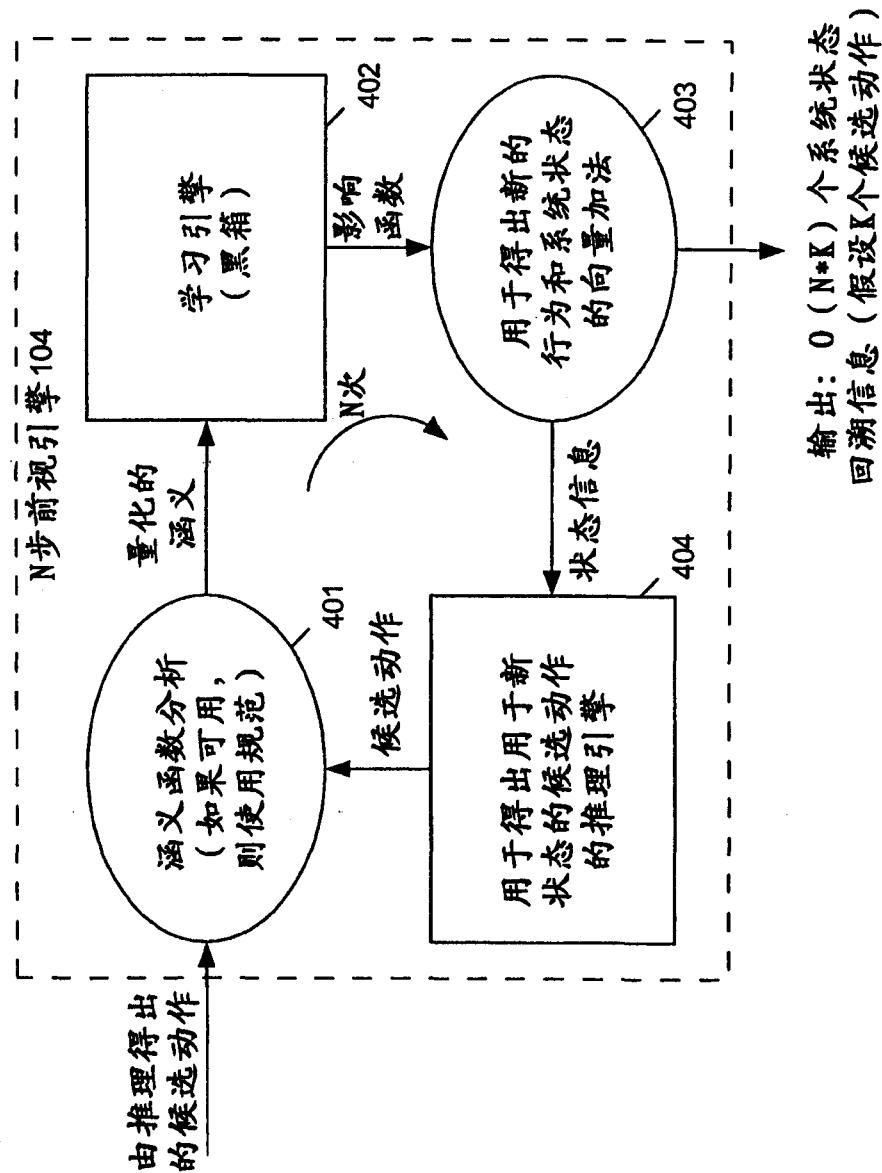


图 5

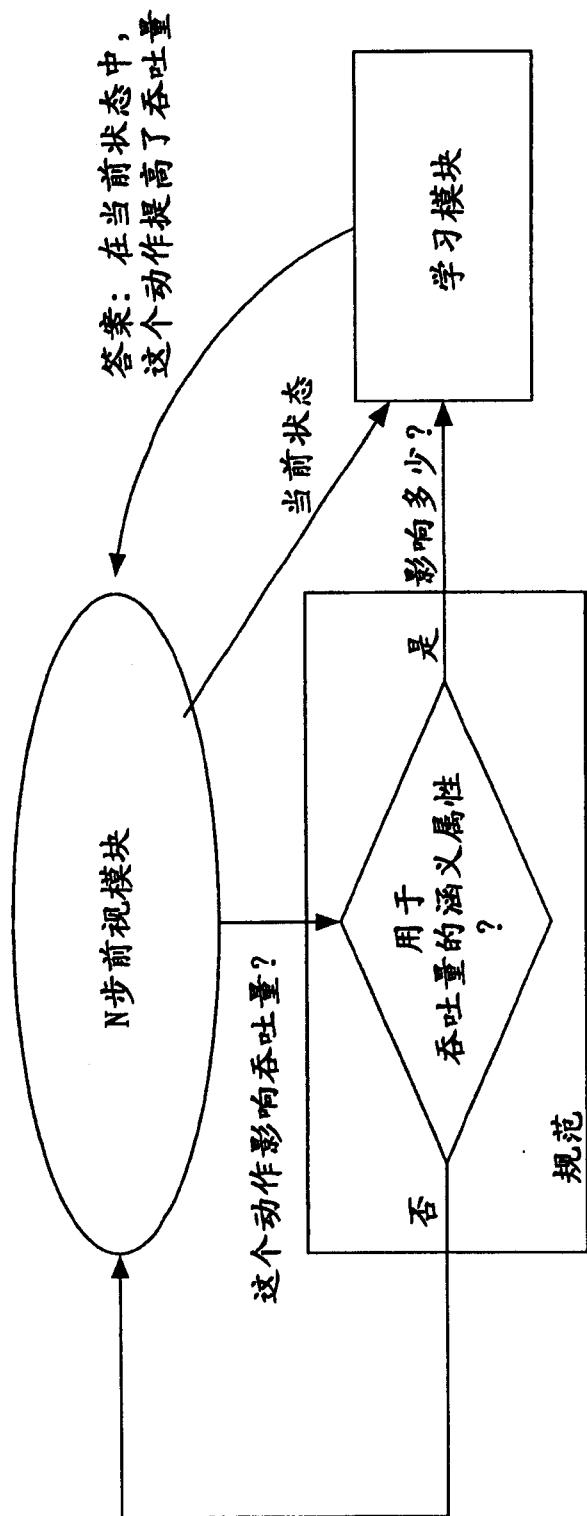


图 6

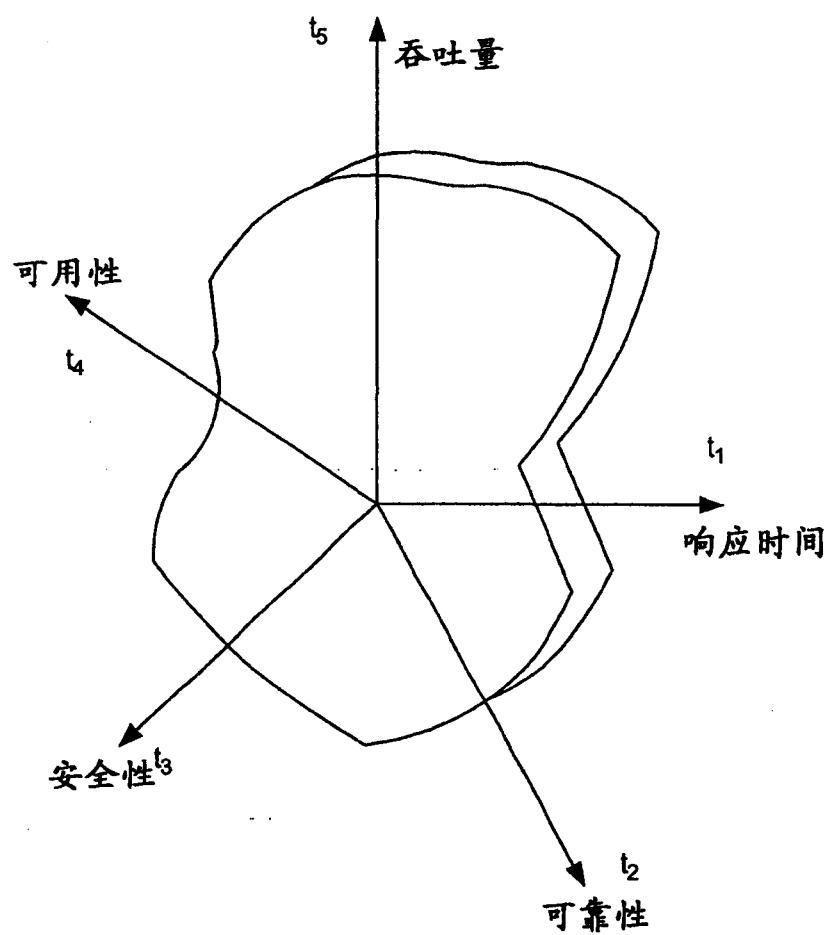


图 7

