



## (12) 发明专利

(10) 授权公告号 CN 102047305 B

(45) 授权公告日 2014. 06. 25

(21) 申请号 200980119736. 8

(51) Int. Cl.

(22) 申请日 2009. 05. 08

G09B 3/00 (2006. 01)

(30) 优先权数据

12/130, 892 2008. 05. 30 US

(56) 对比文件

(85) PCT国际申请进入国家阶段日

2010. 11. 30

CN 102047305 A, 2011. 05. 04,

(86) PCT国际申请的申请数据

PCT/US2009/043387 2009. 05. 08

CN 1617100 A, 2005. 05. 18,

(87) PCT国际申请的公布数据

W02009/148764 EN 2009. 12. 10

US 2006161694 A1, 2006. 07. 20,

(73) 专利权人 美国索尼电脑娱乐有限责任公司

US 7260703 B1, 2007. 08. 21,

地址 美国加利福尼亚州

Pascal Gwosdek. Realtime Optical Flow  
Algorithms on the Cell Processor. 《》. 2008,

(72) 发明人 A. R. 萨勒 E. A. 莱尔纳

审查员 杨雪

R. J. 米卡尔

(74) 专利代理机构 中国专利代理(香港)有限公司 72001

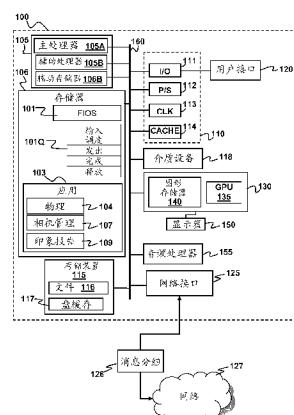
权利要求书4页 说明书18页 附图5页

(54) 发明名称

文件输入 / 输出调度器及其处理方法

(57) 摘要

可以在系统中实现处理进入或来自介质设备的输入或输出(I/O),所述系统具有存储器、处理器单元、以及介质设备,所述处理器单元具有主处理器和辅助处理器,所述辅助处理器具有关联本地存储器。从在处理器单元上运行的应用接收到的输入I/O请求可以根据调度表而受服务。被配置为实现I/O处理的一组处理器可执行指令可以包括介质过滤器层。I/O处理可以替换地包括:从在主处理器上运行的应用接收输入I/O请求;将所述请求插入到在所述主存储器中实施的调度表;根据所述调度表以及一个或多个过滤器实现所述请求,所述一个或多个过滤器中的至少一个是由辅助处理器实现的。



1. 在具有处理器单元、存储器和介质设备的系统中,其中,所述处理器单元包括主处理器和辅助处理器,所述辅助处理器带有关联本地存储器,一种用于处理到所述介质设备的输入 / 来自所述介质设备的输出的方法,所述方法包括:

a) 从在所述处理器上运行的应用接收进入的输入 / 输出请求,以将数据传送到所述介质设备或者传送来自所述介质设备的数据;

b) 计算用于完成所述进入的输入 / 输出请求的预测时间  $T_p$ ;

c) 将所述进入的输入 / 输出请求插入到在所述存储器中实施的调度表,其中,所述调度表内进入的输入 / 输出请求的位置至少部分地取决于所述预测时间  $T_p$ ;

d) 通过将数据从所述介质设备传送到所述本地存储器,并且使用所述辅助处理器对所述本地存储器中的数据进行变换而根据所述调度表服务于所述进入的输入 / 输出请求,其中所述调度表中的后续输入 / 输出请求在所述调度表中的在前输入 / 输出请求完全受服务之后受服务。

2. 权利要求 1 所述的方法,其中,b) 包括:根据性能特征方程来计算  $T_p$ ,所述方程涉及所述介质设备的一个或多个参数以及据所述进入的输入 / 输出请求所确定的一个或多个系数,其中,所述一个或多个参数包括请求开销、头移动速率和吞吐量;其中,所述一个或多个系数包括开销系数、头行进距离和传送数据量。

3. 权利要求 2 所述的方法,其中,所述性能特征方程的形式是: $T_p = Ax+By+Cz$ ,其中 x 是请求开销、y 是头移动速率、z 是吞吐量、A 是开销系数、B 是头行进距离、C 是传送数据量。

4. 权利要求 2 所述的方法,其中,c)包括:对于所述调度表中的多个不同输入 / 输出请求,以不同系数迭代性能特征方程,以确定用于服务于所述输入 / 输出请求的总时间 T。

5. 权利要求 4 所述的方法,其中,c)还包括:对于所述多个输入 / 输出请求的两个或更多个不同顺序,计算总时间 T 的两个或更多个不同的值,并且基于所述总时间 T 的最优值而将所述进入的输入 / 输出请求插入到所述调度表。

6. 权利要求 5 所述的方法,其中,所述总时间 T 的最优值的值是用于所述多个输入 / 输出请求的两个或更多个不同顺序的总时间 T 的最小值。

7. 权利要求 2 所述的方法,其中,b)包括:对于输入 / 输出请求测量所花费的实际时间  $T_a$ ,并且结合所述一个或多个系数使用所述实际时间  $T_a$  来对于一个或多个未知参数的值以迭代方式求解所述性能特征方程。

8. 权利要求 1 所述的方法,其中, d) 还包括:从压缩后的嵌套式存档提取特定资产文件。

9. 权利要求 1 所述的方法,其中,所述系统还包括图形处理器和关联图形存储器,其中, d) 包括:将数据从所述介质设备读入所述图形存储器,将所述数据从所述图形存储器传送到所述本地存储器,使用所述辅助处理器对所述数据执行数据变换以产生变换后的数据,并且将所述变换后的数据传送到所述图形存储器。

10. 权利要求 1 所述的方法,其中, d) 包括:实现数据覆盖,其中,所述介质设备与所述存储器之间传送的一个或多个主数据单元被来自次要数据源的一个或多个次要数据单元所代替。

11. 权利要求 10 所述的方法,其中,所述主数据单元的源和所述次要数据源位于相同介质上。

12. 权利要求 10 所述的方法,其中,所述主数据单元的源和所述次要数据源位于不同介质上。

13. 权利要求 10 所述的方法,其中,所述次要数据源是虚拟数据源。

14. 权利要求 13 所述的方法,其中,所述一个或多个次要数据单元是由回调所指定的。

15. 权利要求 1 所述的方法,其中,所述系统还包括快速存储设备,其中,d)包括:将一个或多个数据单元从所述介质设备预取到所述快速存储设备,并且随后传送来自所述快速存储设备的数据单元。

16. 权利要求 15 所述的方法,其中,所述快速存储设备是硬盘驱动器。

17. 权利要求 16 所述的方法,其中,所述介质设备是高容量光盘驱动器。

18. 权利要求 15 所述的方法,其中,预取一个或多个数据单元包括:在输入 / 输出另外空闲的时间段期间预取所述一个或多个数据单元。

19. 权利要求 18 的方法,其中,预取一个或多个数据单元包括:从所述介质设备预取一个或多个数据单元的第一组,检查所述介质设备的空闲性,并且如果输入 / 输出空闲则从所述介质设备预取一个或多个数据单元的第二组,或者如果输入 / 输出不空闲则延迟预取所述第二组。

20. 权利要求 1 的方法,其中,b)包括:调整用于确定  $T_p$  的一个或多个参数,以仿真比与所述系统关联的所述介质设备更慢且与所述介质设备不同的介质设备的操作。

21. 在具有处理器单元、存储器、快速存储设备、以及介质设备的系统中,一种用于处理到所述介质设备的输入 / 来自所述介质设备的输出的装置,所述装置包括:

a) 用于从在所述处理器上运行的应用接收进入的输入 / 输出请求,以将数据传送到所述介质设备或者传送来自所述介质设备的数据的部件;

b) 用于计算用于完成进入的输入 / 输出请求的预测时间  $T_p$  的部件;

c) 用于将所述进入的输入 / 输出请求插入到在所述存储器中实施的调度表的部件,其中,所述调度表内进入的输入 / 输出请求的位置至少部分地取决于所述预测时间  $T_p$ ;以及

d) 用于根据所述调度表实现所述进入的输入 / 输出请求的部件,其中所述调度表中的后续输入 / 输出请求在所述调度表中的在前输入 / 输出请求被完全实现之后而被实现,其中,b)包括:用于调整用于确定  $T_p$  的一个或多个参数,以仿真比与所述系统关联的所述介质设备更慢且与所述介质设备不同的介质设备的操作的部件。

22. 在具有主处理器、带有关联本地存储器的辅助处理器、主存储器、以及介质设备的系统中,一种用于处理到所述介质设备的输入 / 来自所述介质设备的输出的方法,所述方法包括:

a) 从在所述主处理器上运行的应用接收进入的输入 / 输出请求,以将数据传送到所述介质设备或者传送来自所述介质设备的数据;

b) 将所述进入的输入 / 输出请求插入到在所述主存储器中实施的调度表;以及

c) 根据所述调度表和一个或多个过滤器服务于所述进入的输入 / 输出请求,其中所述调度表中的后续输入 / 输出请求在所述调度表中的在前输入 / 输出请求完全受服务之后受服务,其中,所述一个或多个过滤器中的至少一个是由所述辅助处理器实现的。

23. 在具有处理器单元、存储器以及介质设备的系统中,一种用于处理到所述介质设备的输入 / 来自所述介质设备的输出的方法,所述方法包括:

a) 从在所述处理器上运行的应用接收进入的输入 / 输出请求,以将数据传送到所述介质设备或者传送来自所述介质设备的数据;

b) 计算用于完成所述进入的输入 / 输出请求的预测时间  $T_p$ ;

c) 将所述进入的输入 / 输出请求插入到在所述存储器中实施的调度表,其中,所述调度表内进入的输入 / 输出请求的位置至少部分地取决于所述预测时间  $T_p$ ;以及

d) 根据所述调度表实现所述进入的输入 / 输出请求,其中所述调度表中的后续输入 / 输出请求在所述调度表中的在前输入 / 输出请求被完全实现之后而被实现,其中,d) 包括:从压缩后的嵌套式存档提取特定资产文件。

24. 在具有处理器单元、存储器以及介质设备的系统中,一种用于处理到所述介质设备的输入 / 来自所述介质设备的输出的方法,所述方法包括:

a) 从在所述处理器上运行的应用接收进入的输入 / 输出请求,以将数据传送到所述介质设备或者传送来自所述介质设备的数据;

b) 计算用于完成所述进入的输入 / 输出请求的预测时间  $T_p$ ;

c) 将所述进入的输入 / 输出请求插入到在所述存储器中实施的调度表,其中,所述调度表内进入的输入 / 输出请求的位置至少部分地取决于所述预测时间  $T_p$ ;以及

d) 根据所述调度表实现所述进入的输入 / 输出请求,其中所述调度表中的后续输入 / 输出请求在所述调度表中的在前输入 / 输出请求被完全实现之后而被实现,其中,d) 包括:实现数据覆盖,其中,在所述介质设备与所述存储器之间传送的一个或多个主数据单元被来自次要数据源的一个或多个次要单元所替代。

25. 权利要求 24 所述的方法,其中,所述主数据单元的源和所述次要数据源位于相同介质上。

26. 权利要求 24 所述的方法,其中,所述主数据单元的源和所述次要数据源位于不同介质上。

27. 权利要求 24 所述的方法,其中,所述次要数据源是虚拟数据源。

28. 权利要求 24 所述的方法,其中,所述一个或多个次要数据单元是由回调所指定的。

29. 在具有处理器单元、存储器、快速存储设备、以及介质设备的系统中,一种用于处理到所述介质设备的输入 / 来自所述介质设备的输出的方法,所述方法包括:

a) 从在所述处理器上运行的应用接收进入的输入 / 输出请求,以将数据传送到所述介质设备或者传送来自所述介质设备的数据;

b) 计算用于完成所述进入的输入 / 输出请求的预测时间  $T_p$ ;

c) 将所述进入的输入 / 输出请求插入到在所述存储器中实施的调度表,其中,所述调度表内进入的输入 / 输出请求的位置至少部分地取决于所述预测时间  $T_p$ ,其中所述输入 / 输出请求包括对在预料到对文件的未来需要时从所述介质设备预取一个或多个数据单元的请求;以及

d) 根据所述调度表实现所述进入的输入 / 输出请求,其中所述调度表中的后续输入 / 输出请求在所述调度表中的在前输入 / 输出请求被完全实现之后而被实现,其中,d) 包括:将所述一个或多个数据单元从所述介质设备预取到所述快速存储设备,以及如果所述一个或多个数据单元被所述系统需要则随后传送来自所述快速存储设备的数据单元,或者如果所述一个或多个数据单元不被所述系统需要则取消对预取所述一个或多个数据单元的请

求。

30. 权利要求 29 所述的方法,其中,所述快速存储设备是硬盘驱动器。
31. 权利要求 30 所述的方法,其中,所述介质设备是高容量光盘驱动器。
32. 权利要求 31 所述的方法,其中,预取一个或多个数据单元包括:在输入 / 输出另外空闲的时间段期间预取所述一个或多个数据单元。
33. 权利要求 31 的方法,其中,预取一个或多个数据单元包括:从所述介质设备预取一个或多个数据单元的第一组,检查所述介质设备的空闲性,并且如果输入 / 输出空闲则从所述介质设备预取一个或多个数据单元的第二组,或者如果输入 / 输出不空闲则延迟预取所述第二组。
34. 在具有处理器单元、存储器、快速存储设备、以及介质设备的系统中,一种用于处理到所述介质设备的输入 / 来自所述介质设备的输出的方法,所述方法包括:
  - a) 从在所述处理器上运行的应用接收进入的输入 / 输出请求,以将数据传送到所述介质设备或者传送来自所述介质设备的数据;
  - b) 计算用于完成所述进入的输入 / 输出请求的预测时间  $T_p$ ;
  - c) 将所述进入的输入 / 输出请求插入到在所述存储器中实施的调度表,其中,所述调度表内进入的输入 / 输出请求的位置至少部分地取决于所述预测时间  $T_p$ ;以及
  - d) 根据所述调度表实现所述进入的输入 / 输出请求,其中所述调度表中的后续输入 / 输出请求在所述调度表中的在前输入 / 输出请求被完全实现之后而被实现,其中,b) 包括:调整用于确定  $T_p$  的一个或多个参数,以仿真比与所述系统关联的所述介质设备更慢且与所述介质设备不同的介质设备的操作。

## 文件输入 / 输出调度器及其处理方法

[0001] 优先权声明

[0002] 该申请要求 2008 年 5 月 30 日提交的共同受让美国专利申请 12/130,892 的优先权之利益，其整个公开在此引入以供参考。

### 技术领域

[0003] 本发明实施例涉及计算机游戏以及有关的应用，更具体地说，涉及计算游戏以及有关应用中的文件输入 / 输出(I/O)管理。

### 背景技术

[0004] 很多软件应用(例如视频游戏)包括文件输入输出(I/O)调度器，以使得应用内的介质存取更高效并且可靠。文件 I/O 调度器(“FIOS”)是中间件层，用于存取带有若干部分的文件，其包括调度器和可选 I/O 过滤器层。调度器典型地被设计为优化地对 I/O 请求排序，从而它们在服从任意最终期限和优先级约束的最短可能时间内完成。过滤器层可以提供额外服务，例如解压缩或缓存。

[0005] 很多不同游戏组件需要对存储介质中的文件的 I/O 存取。音频组件加载音频文件；游戏玩乐引擎加载级别清晰度；图形组件加载纹理地图和模型；电影组件加载音频视频文件；以及子系统加载大型 WAD 文件。介质可以是游戏传递介质(例如光盘(通用介质盘(UMD)、致密盘(CD)、数字视频盘(DVD)、蓝光盘(BD)等))、中间存储介质(例如硬盘)、或随着平台演进的其它介质类型。

[0006] 单一应用(例如视频游戏)典型地具有多个组件，每一组件具有其自身的 I/O 需求。某些需要对介质的流式存取，其中，I/O 系统将文件中的数据流送到组件，从而该组件可以将流送来的数据连续呈现在游戏机上。例如，音频组件典型地流送用于音轨回放的音频文件；电影组件流送音频视频内容，用于播放器回放电影。其它组件仅需要非流式存取，其中，它们成块地从文件检索数据，以供组件处理。这些组件不需要稳固的数据流，但块传递时序一般是严格的。如果应用未在需要时接收到数据，则性能可能遭损。

[0007] 视频游戏必须一般执行相当大量 I/O。游戏事件一般是按时间驱动的，具有必须被满足的最终期限。例如，如果玩家正从点 A 行进到点 B，则游戏需要具有用于在玩家达到点 B 之前加载的与点 B 关联的数据的文件。游戏组件可以使用低级别 I/O 原语来从介质检索数据，但当多个组件同时需要来自同一设备的数据时，这些原语不处理设备竞争。拥塞的 I/O 请求能够中断流送数据流，或者禁止关键数据块在需要之时到达组件。较高级别系统(例如 989Sound's Streamsafe)至少提供对文件的可靠流存取，从而流送的数据不受中断。遗憾的是，这些系统仅分配单个数据流，而未很好地处理非流数据的竞争。常规系统未提供对于正常数据存取解决竞争的标准层。

[0008] 因为应用的组件典型地不知道彼此的 I/O 活动，所以它们的 I/O 请求通常导致带有大量过度搜寻(对数据的前后搜索)的异常低效的 I/O 模式。随着游戏增长得较大，过度搜寻和低效数据检索增加。用于分布式介质(例如 CD)的盘主控(disc mastering)可以通

过各技术(例如频繁复制所存取的数据块,并且穿过盘来发布它们,从而拷贝将总是靠近,无论存储设备的读取头在何处)帮助避免这种无效性中的某些。文件系统并非总是显示数据在物理上位于盘上的何处,尽管 I/O 系统不能总是最佳使用数据。

[0009] 在此情况下提出本发明实施例。

## 发明内容

[0010] 根据本发明实施例,一种用于处理去往或来自介质设备的输入或输出(I/O)的方法可以得以实现在具有处理器单元、存储器和介质设备的系统中。可以从在所述处理器上运行的应用接收输入 I/O 请求,以将数据传送到介质设备或者传送来自介质设备的数据。可以计算出用于完成输入 I/O 请求的预测时间  $T_p$ 。输入 I/O 请求可以插入到在所述存储器中实施的调度表。所述调度表内输入 I/O 请求的位置可以至少部分地取决于预测时间  $T_p$ 。输入 I/O 请求可以根据调度表而受服务。

[0011] 在某些实施例中,一组处理器可执行指令可以实施在所述存储器中。所述指令可以被配置为当执行时实现上述方法。所述一组处理器可执行指令可以包括一个或多个介质过滤器层。所述介质过滤器层可以包括解存档器层、RAM 缓存层、调度器缓存(例如硬盘驱动器(HDD)缓存)层、覆盖层、分类缓存层、数据变换层、或仿真数据层。

[0012] 在某些实施例中,所述预测时间  $T_p$  可以根据性能特征方程(PCE)而得以计算,所述 PCE 涉及所述介质设备的一个或多个参数以及根据所述输入 I/O 请求而确定的一个或多个系数。通过示例的方式,并且不失一般性,所述参数包括请求开销  $x$ 、头移动速率  $y$  和吞吐量  $z$ 。在该示例中,所述系数可以包括开销系数  $A$ 、头行进距离  $B$  和传送数据量  $C$ 。所述性能特征方程可以是这样的形式 : $T_p = Ax+By+Cz$ 。

[0013] 在某些实施例中,用于 I/O 请求的所述 PCE 系数可以是基于所述 I/O 请求以及初始介质状态而确定的,其可以包含关于影响性能的设备的状态的一条或多条信息。通过示例的方式,并且不失一般性,所述一条或多条信息可以包括设备的准备状态、以及所存取的最近逻辑块地址(LBA)。在某些实施例中,根据 I/O 请求和初始介质状态计算所述 PCE 系数可以返回更新后的介质状态,其可以用作另一 PCE 计算的初始介质状态。

[0014] 在某些实施例中,所述 PCE 系数可以与 PCE 中的未知量的值组合,以计算用于任何给定的 I/O 请求的预测时间  $T_p$ 。用于 I/O 请求的排序列表的总时间  $T$  可以随后通过如下方式得以计算,即 :对于所述列表中的每一 I/O 请求迭代所述 PCE 和介质状态,并且对所得时间  $T_p$  求和。该过程可以对于 I/O 请求的所述排序列表的某些或所有排列而重复。所述列表的优选排序可以基于对于每一排序而计算出的时间  $T$  以及可能的另外准则而选自所述多个排列当中。在某些实施例中,优选排序可以是产生最长时间  $T$  的排列。可以影响排序的额外准则可以包括(但不限于) I/O 请求优先级、流缓冲器状态、以及等待时间需求。

[0015] 在某些实施例中,对于任何 I/O 请求所花费的实际时间  $T_a$  可以是结合 PCE 系数而测量并且使用的,从而对于未知量迭代地求解 PCE。这样可以允许模型以实际驱动器性能而得以恒定地更新。

[0016] 在某些实施例中,实现所述 I/O 请求的操作还包括 :从压缩后的嵌套式存档提取特定数据。

[0017] 在某些实施例中,所述处理单元可以包括主处理器、辅助处理器,所述辅助处理器

具有关联本地存储器。在这些实施例中,实现所述 I/O 请求的步骤可以包括:将数据从所述介质设备传送到所述本地存储器,并且使用所述辅助处理器对所述本地存储器中的数据进行变换。

[0018] 在某些实施例中,所述系统可以还包括图形处理器和关联图形存储器。在这些实施例中,实现所述 I/O 请求的步骤可以包括:将数据从所述介质设备传送到所述图形存储器,将所述数据从所述图形存储器传送到所述本地存储器,使用所述辅助处理器对所述数据执行数据变换以产生变换后的数据,将变换后的数据传送回到所述图形存储器。

[0019] 在某些实施例中,实现所述 I/O 请求的操作可以包括:实现数据覆盖,其中,所述介质设备与所述存储器之间传送的一个或多个主数据单元被来自辅数据源的一个或多个次要数据单元所代替。所述主数据单元的源和所述次要数据源可以位于相同介质上或不同介质上。所述次要数据源可以是虚拟数据源。所述次要数据单元可以通过回调而得以指定。

[0020] 在某些实施例中,所述系统可以还包括快速存储设备。在此情况下,实现所述 I/O 请求的步骤可以包括:将一个或多个数据单元从所述介质设备预取到所述快速存储设备,随后传送来自所述快速存储设备的数据单元。通过示例的方式,所述快速存储设备可以是硬盘驱动器,介质设备可以是高容量光盘驱动器,例如蓝光盘驱动器。所述数据单元可以在 I/O 另外空闲的时间段期间得以预取。在某些实施例中,所述预取步骤可以包括:从所述介质设备预取一个或多个数据单元的第一组,检查所述慢介质设备的空闲性,并且如果 I/O 空闲则从所述慢介质设备预取一个或多个数据单元的第二组,或者如果 I/O 不空闲则延迟预取所述第二组。

[0021] 在某些实施例中,计算所述预测时间  $T_p$  的步骤可以包括:调整所述参数,以仿真比与所述系统关联的实际介质设备更慢的介质设备的操作。

[0022] 根据本发明实施例,一种用于处理去往或来自介质设备的输入或输出(I/O)的方法可以得以实现在具有主处理器、带有关联本地存储器的辅助处理器、主存储器和介质设备的系统中。所述方法可以包括:a)从在所述处理器上运行的应用接收输入 I/O 请求,以将数据传送到所述介质设备或者传送来自所述介质设备的数据;b)将所述输入 I/O 请求插入到在所述主存储器中实施的调度表;以及 c)根据所述调度表和一个或多个过滤器实现所述输入 I/O 请求,其中,所述过滤器中的至少一个是由所述辅助处理器实现的。

## 附图说明

[0023] 通过结合附图考虑以下详细描述,可以容易地理解本发明的教导。

[0024] 图 1 是根据本发明实施例的实现文件 I/O 系统(FIOS)的系统的框图。

[0025] 图 2 是根据本发明替换实施例的实现 FIOS 的替换系统的框图。

[0026] 图 3 是根据本发明实施例的文件输入 / 输出系统(FIOS)软件的框图。

[0027] 图 4 是示出根据本发明实施例的在 FIOS 中的操作流程的数据流程图。

[0028] 图 5 是示出根据本发明实施例的在 FIOS 中使用的软件调度器的示例的流程图。

[0029] 图 6 是示出根据本发明实施例的在 FIOS 中预取操作的示例的示意图。

[0030] 图 7A 是示出根据本发明实施例的使用带有关联本地存储器的辅助处理器的 I/O 缓冲的示例的流程图。

[0031] 图 7B 是示出根据本发明实施例的使用带有关联本地存储器的辅助处理器的 I/O 缓冲和数据变换的示例的流程图。

[0032] 图 8 是示出可以结合本发明实施例而使用的用于嵌套式存档的数据结构的框图。

[0033] 图 9 是示出根据本发明实施例的解存档器的流程图。

## 具体实施方式

[0034] 虽然以下详细描述为了说明而包含很多特定细节,但本领域技术人员应理解,对于以下细节的很多变形和改动是在本发明范围内的。相应地,以下描述的本发明实施例的示例是在不失一般性的情况下阐述的,并且不将限制施加于本发明。

[0035] 本发明实施例可以围绕文件 I/O 调度器(FIOS)而得以实现,FIOS 提供用于系统的所有 I/O 穿过的中心层。FIOS 可以包括调度器,其对 I/O 请求进行排序,并且确定最有效地服务于 I/O 请求的顺序。

[0036] 通过示例的方式,以计算机实现的系统 100 可以被配置为实现图 1 所示的根据本发明实施例的文件 I/O 调度器(FIOS)。通过示例的方式,并且不失一般性,系统 100 可以实现为个人计算机、视频游戏控制台、个人数字助理、或其它数字设备,适合于实践本发明实施例。系统 100 可以包括处理单元 105 和主存储器 106,主存储器 106 耦合到处理 105。CPU 105 被配置为运行软件应用,并且可选地运行操作系统。本发明某些实施例可以利用特定类型的处理器架构,其中, CPU 105 包括主处理器 105A 和辅助处理器 105B,辅助处理器 105B 具有其自身的关联本地存储器 106B。其中,这样的处理器架构的一个例子是单元处理器(Cell Processor)。单元处理器架构的示例详细描述于例如单元宽带引擎架构(Cell Broadband Engine Architecture)中,版权归属 International Business Machines Corporation、Sony Computer Entertainment Incorporated、Toshiba Corporation,2005 年 8 月 8 日,其拷贝可以在 <http://cell.scei.co.jp/> 下载,其完整内容在此引入以供参考。

[0037] 主存储器 106 可以存储应用和数据,以供 CPU 105 使用。存储器 106 可以是集成电路的形式,例如 RAM、DRAM、ROM 等。计算机程序 101 可以存储于存储器 106 中,形式为可以在处理器 105 上执行的指令。程序 101 的指令可以被配置为其中实现具有以下所描述的特定特征的文件输入 / 输出系统(FIOS)。存储器 106 可以包含 I/O 队列 101Q,例如形式是堆栈或队列,用于输入、调度、发出、完成以及释放 FIOS 程序 101 所使用的 I/O 请求。这些队列的示例也描述如下。

[0038] 通过示例的方式,FIOS 程序 101 可以包括各指令,用于 :a)从应用 103 接收涉及介质设备 118 的输入 I/O 请求,b)计算用于完成 I/O 请求的预测时间  $T_p$ ,c)至少部分地基于预测时间  $T_p$  将输入 I/O 请求插入到存储器 106 中的调度表的位置,d)根据调度表实现 I/O 请求。

[0039] FIOS 程序 101 可以结合被配置为实现交互式环境的一条或多条指令而操作。通过示例的方式,这些指令可以是主程序 103 (例如视频游戏程序)的子程序或可调用函数。替换地,主程序 103 可以是用于与虚拟世界进行接口的程序。主程序 103 可以被配置为将来自相机 POV 的一部分仿真环境的场景显示在视频显示器上,并且响应于相机 POV 在用户与仿真环境的交互期间沿着相机路径的移动而随着相机 POV 改变来改变场景。主程序可以包

括用于物理仿真 104 和相机管理 107 的指令。主程序 103 可以调用 FIOS 程序 101、物理仿真指令 104、相机管理指令 107 以及广告印象报告指令 109，例如作为函数或子程序。

[0040] 客户机系统 100 也可以包括公知的支持功能 110，例如输入 / 输出(I/O)元件 111、电源(P/S)112、时钟(CLK)113 和缓存 114。客户机设备 100 可以还包括快速存储设备 115，例如硬盘驱动器，其为应用和数据提供非易失性存储。其中，快速存储设备 115 可以用于临时或者长期存储从较慢介质设备 118 检索到的文件 116。快速存储设备 115 上的文件 116 可以另外来自除了较慢介质设备 118 之外的源。例如，文件 116 可以包括但不限于操作系统文件、由应用所创建的临时文件、用户数据，例如相片 / 音频 / 视频、下载的内容、以及更多。通过示例的方式，存储设备 115 可以是固定盘驱动器，可拆卸盘驱动器、闪速存储器设备、带式驱动器。较慢介质设备 118 可以是高容量光盘驱动器，例如 CD-ROM 驱动器、DVD-ROM 驱动器、高清晰度数字多功能盘(HD-DVD)驱动器、蓝光盘驱动器、UMD 驱动器、或其它光学存储设备。来自介质设备 118 的预取出的文件 116 可以临时存储于硬件缓存中的存储设备 115 中，以便快速加载到存储器 106。

[0041] 一个或多个用户输入设备 120 可以用于将用户输入从一个或多个用户传递到系统 100。通过示例的方式，用户输入设备 120 中的一个或多个可以经由 I/O 元件 111 钮合到客户机设备 100。合适的输入设备 120 的示例包括键盘、鼠标、操纵杆、触摸板、触摸屏、光笔、相机或摄像机、和 / 或麦克风。客户机设备 100 可以包括网络接口 125，用于促进经由电子通信网络 127 进行的通信。网络接口 125 可以被配置为在局域网和广域网(例如互联网)上实现有线通信或无线通信。系统 100 可以在网络 127 上经由一个或多个消息分组 126 来发送并且接收数据和 / 或对文件的请求。

[0042] 系统 100 可以还包括图形子系统 130，图形子系统 130 可以包括图形处理单元(GPU)135 和图形存储器 140。图形存储器 140 可以包括显示存储器(例如帧缓冲器)，用于存储输出图像的每一像素的像素数据。图形存储器 140 可以与 GPU 135 集成在同一设备中，作为分离设备而与 GPU 135 连接，并且 / 或者实现在存储器 106 内。像素数据可以直接从 CPU 105 提供给图形存储器 140。替换地，CPU 105 可以向 GPU 135 提供定义期望输出图像的数据和 / 或指令，GPU 135 据此可以生成一个或多个输出图像的像素数据。定义期望输出图像的数据和 / 或指令可以存储于存储器 106 和 / 或图形存储器 140 中。在实施例中，GPU 135 可以(例如通过合适的编程或硬件配置)配置有 3D 呈现能力，用于据定义场景的几何、光照、阴影、纹理、移动和 / 或相机参数的指令和数据来生成输出图像的像素数据。GPU 135 可以还包括能够执行阴影化(shader)程序的一个或多个可编程执行单元。

[0043] 图形子系统 130 可以从图形存储器 140 周期性地输出图像的像素数据，以显示在视频显示设备 150 上。视频显示设备 150 可以是能够响应于来自系统 100 的信号而显示可视信息的任何设备，包括 CRT 显示器、LCD 显示器、等离子体显示器、以及 OLED 显示器。计算机系统 100 可以向显示设备 150 提供模拟信号或数字信号。通过示例的方式，显示器 150 可以包括阴极射线管(CRT)或平板屏幕，其显示文本、数字、图形符号或图像。此外，显示器 150 可以包括一个或多个音频扬声器，其产生听觉声音或另外可检测到的声音。为了促进生成这种声音，客户机设备 100 可以还包括音频处理器 155，其适用于据 CPU 105、存储器 106 和 / 或存储装置 115 所提供的指令和 / 或数据而生成模拟音频输出或数字音频输出。

[0044] 包括 CPU 105、存储器 106、支持功能 110、数据存储设备 115、介质设备 118、用户输

入设备 120、网络接口 125 和音频处理器 155 的系统 100 的各组件可操作地经由一条或多条数据总线 160 而彼此连接。这些组件可以通过硬件、软件或固件或它们中的两个或更多个的某种组合而得以实现。

[0045] 本发明某些实施例可以利用单元处理器架构或相似的处理器架构。图 2 示出根据本发明实施例的被配置为实现 FIOS 的单元处理器 200 的示例。单元处理器 200 包括主存储器 202、单个功率处理器元件(PPE)204 和八个协作处理器元件(SPE)206。通过示例的方式，PPE 204 可以包括带有关联缓存的 64 比特 PowerPC 处理器单元(PPU)。某些实现方式(例如 CBEA 顺从的系统)可以包括将矢量多媒体扩展单元包括在 PPE 204 中。PPE 204 可以是通用处理单元，其可以访问系统管理资源(例如存储器保护表)。硬件资源可以明确映射为 PPE 204 可见的真实地址空间。因此，PPE 204 可以通过使用适当的有效地址值来直接对任何这些资源进行寻址。PPE 204 的主要功能是为不同的 SPE 206 管理并且分配任务。PPU 可以执行 FIOS 程序 205 的编码指令。

[0046] SPE 206 是比 PPE 204 不太复杂的计算单元，这是由于它们不需执行任何系统管理功能。每一 SPE 206 包括处理器单元，有时称为协作处理器单元(SPU)和关联本地存储(LS)。SPE 206 可以通常具有单指令多数据(SIMD)能力，并且典型地处理数据并且发起任何所需的数据传送(服从 PPE 204 所设置的访问特性)，以便执行它们的所分配的任务。SPE 206 可以在其本地存储中存储实现 FIOS 程序 205 的各部分的指令 207。SPE 206 的目的在于启用需要更高计算单元密度并且可以有效使用所提供的指令集的应用。虽然该示例中示出八个 SPE，但单元处理器 200 可以配置有任何数量的 SPE。关于图 2，存储器 202、PPE 204 和 SPE 206 可以在环形元件互连总线 210 上彼此进行通信，并且与 I/O 设备 208 进行通信。存储器 202 可以由 PPE 204 和 SPE 206 经由存储器接口控制器 MIC 来存取。

[0047] 存储器 202 可以包含 I/O 队列 203，例如输入、调度、发出、完成、释放和预取队列，如下所述。存储器 202 也可以包含具有与在此描述的 FIOS 程序 101 共同的特征的 FIOS 程序 209 的各部分。SPE 206 中的至少一个可以在其本地存储中包括代码 207，其被配置为实现如下所述的数据解压缩。PPE 204 可以包括内部缓存 L1 和外部缓存 L2。PPE 204 可以将 FIOS 程序 205 的各部分存储在其内部缓存 L1 中。FIOS 程序 205 和以 SPE 实现的文件传送指令 207 或其各部分也可以存储于存储器 202 中，以便当需要时由 SPE 206 和 PPE 204 来存取。

[0048] 通过示例的方式，FIOS 程序 101/205 可以包括介质堆栈，用于促进与硬件(例如存储设备 115)的接口。介质堆栈可以实现为图 3 所描述的那样。具体地说，介质堆栈 300 可以包括调度器 302、一个或多个介质过滤器层 304、设备介质层 306、处理器文件系统(FS)读取层 308 和硬件层 310。设备介质层 306(有时称为介质存取层)可以被配置为响应于从上面介质过滤器层 304 接收到的介质请求，检索所请求的数据，然后将其应答发送回至堆栈。介质过滤器层 304 可以包括解存档器层 301、RAM 缓存层 303、调度器缓存层 305、覆盖层 307、分类缓存层 309、一个或多个数据变换层 311、以及仿真数据层 313、还有速度仿真层 315。

[0049] 解存档器 301 可以用于促进从压缩后的存档提取特定资产文件。RAM 缓存层 303 可以用于实现将数据缓存在例如主存储器 106、202 中。调度器缓存 305 可以是简单的盘对盘缓存，用于临时存储来自较慢源(例如光盘)的数据。如在此所使用的那样，“调度器缓存”

指代在存储介质中的临时数据存储,其比介质设备 118 更快地进行存取。并非需要预取调度器缓存 305 中的所有数据;某些数据可以按需取得,并且拷贝到缓存。

[0050] 通过示例的方式,而非通过限制的方式,调度器缓存层 305 可以利用快速存储介质 115 来提供这种临时存储。在快速存储介质是硬盘驱动器(HDD)的特定情况下,调度器缓存 305 有时称为 HDD 缓存。

[0051] 调度器缓存 305 可以作为单个文件或多个文件而得以保存。此外,调度器缓存 305 的内容无需是全部文件。多文件缓存可以通过删除某些单独文件而得以部分地清洗,从而当需要时智能地释放盘空间,而不牺牲太多数据。通过对照,单文件缓存典型地可以仅被截断或者完全删除。单文件缓存可以提供比多文件缓存更高的性能,因为多文件缓存典型地需要另外的簿记工作(在主机文件系统自身内部),这可能需要额外的 I/O。

[0052] 覆盖层 307 可以用于允许在文件系统级别任意覆盖文件和目录,如下所述。分类缓存层 309 可以用于对处理器单元 105 或单元处理器 200 的 O/S 可能未正确地缓存(例如文件存在性、大小、以及位置)的数据进行缓存。数据变换层 311 可以对读自或写入介质设备 118 的数据实现数据变换,例如加密、解密、压缩或解压缩。仿真数据层 313 可以例如用于生成多个零或随机数,以用于需要这种仿真数据的 I/O 请求。速度仿真层 315 可以用在例如软件开发期间,以对介质设备的不同速度进行仿真,如下所述。

[0053] 可以关于如下所述的图 4 来理解调度器 302 的操作。从客户机应用的观点来看,I/O 请求 403 可以是以相对直接的方式而受服务的。例如,应用 401(例如视频游戏)内的线程 402 可以凭借调用函数(例如 `readFile()`)通过 FIOS 101/205 来请求 I/O。该函数可以指定 FIOS 请求 403 的优先级以及该请求应完成的最终期限。该函数也可以将指针提供给缓冲器 405。这种实现方式可以按原子方式来分配用于 I/O 请求 403 的 I/O 操作结构,并且将新分配的操作移动到输入队列 404。通过示例的方式,输入队列可以是先入先出(FIFO)队列,其中,队列 404 中放入的最先 I/O 操作是由调度器 302 调度的最先操作。将请求 403 插入到输入队列 404 的操作可以通过原子操作而得以实现,以防止当请求异步 I/O 时线程受阻。在某些实施例中,输入队列 404 可以是原子堆栈的形式,其可以由辅助处理器 105B 通过原子交换来填充。

[0054] 通过示例的方式,而非通过限制的方式,如果 I/O 请求 403 为读取操作,则客户机应用 401 可以提供缓冲器,以接收从介质设备检索到的数据。当 I/O 请求完成时,客户机应用可以从其所提供的缓冲器读取所检索的数据。当客户机终结于操作结构时,客户机可以解除分配操作结构,从而,其资源可用于后来的 I/O 用途。

[0055] I/O 请求 403 可以在调度器尚未激活的情况下激活调度器 302,并且可以将用户可见的句柄返回到结构,从而应用 401 具有对操作的存取。应用 401 于是可以等待 I/O 请求 403 完成。例如,客户机应用可以使用原子方法(例如 `isDone()`)来周期性地进行轮询。替换地,当 I/O 请求完成时,客户机应用可以等待调度器 302 调用的回调函数。

[0056] 在 I/O 请求已经插入到输入队列 404 之后,FIOS 程序 101/205 可以调用调度器 302 来执行调度表插入 407。可以参照图 5 来理解调度器所实现的各操作的序列。如果没有 I/O 要处理,则调度器 302 可以休眠(即保持不激活),如在 502 所指示的那样。当新 I/O 操作进入到输入队列时,调度器唤醒,以对其进行处理。一旦调度器唤醒过来(或者已经唤醒),其就注意队列中的新操作,并且将操作移动到调度队列 406 中的适当位置。

[0057] 为了确定 I/O 请求 403 的队列位置, 调度器 302 可以可选地例如通过询问设备介质层 306 或 FIOS 堆栈 300 中的另一层来确定介质设备的当前状态, 如在 503 所指示的那样。状态数据可以根据 FIOS 堆栈 300 所处理的介质设备的类型以及堆栈中出现的各种层而变化。各示例包括最近近存取的逻辑块地址 (LBA)、RAM 缓存之最近存取的路径和偏移、当前盘层、头位置、流送模式、以及非流送模式。

[0058] 根据本发明实施例, 调度器 302 可以是基于介质设备性能模型 302A。不同于现有 I/O 调度器的是, 驱动器模型可以在确定请求 403 的最佳调度中将关于介质设备 118 之性能的数据纳入考虑。使用驱动器模型来调度 I/O 请求可以与在生成 CD 的主控过程的逆过程相比较。性能模型 302A 可以被配置为将诸如开销、盘移动时间、读取时间等之类的因素纳入考虑。性能模型 302A 可以对介质设备 118 的任意复杂特性(例如吞吐量、激光器摆动、读取头移动、层改变以及请求开销)进行建模。性能模型 302A 也可以将 I/O 请求中所涉及的介质设备 118 所读取或者写入的特定介质的其它参数纳入考虑。例如, 性能模型 302A 可以考虑设备是正从单层盘还是多层盘(例如双层盘, 比如蓝光 DMD)进行读取。

[0059] 调度器 302 也可以可选地查看新 I/O 操作时序要求、优先级、以及可能的效率, 如在 504 所指示的那样。调度器可以被配置为计算预测时间  $T_p$ , 以完成输入 I/O 请求, 如在 506 所指示的那样。根据优选实施例, 调度器 302 可以根据具有与介质设备 118 有关的变量参数以及与 I/O 请求 403 有关的系数的性能特征方程 (PCE) 来确定用于执行 I/O 请求 403 的预测时间  $T_p$ 。

[0060] PCE 可以具有一般形式:

$$[0061] T_p = Af_1(X) + Bf_2(Y) + Cf_3(Z) + Df_4(W) + Ef_5(V) + \dots,$$

[0062] 其中, A、B、C、D 以及 E 是系数, x、y、z、w 和 v 表示与介质设备的性能特征有关的参数,  $f_1(x)$ 、 $f_2(y)$ 、 $f_3(z)$ 、 $f_4(w)$ 、 $f_5(v)$ …是 x、y、z、w 和 v 的函数。通过示例的方式, 而非通过限制的方式, PCE 可以是线性方程, 形式是一个或多个项之和, 其中, 每一项是系数 (A、B、C 等) 乘以表示参数值的变量 (x, y, z 等)。这种方程可以具有以下形式:

$$[0063] T_p = Ax + By + Cz + Dw + Ev + \dots$$

[0064] 该方程可以具有任意数量的项。各变量可以对应于影响用于执行所要求的任务的时间的介质设备的任何参数。所述参数的示例包括但不限于开销、搜寻时间、以及数据传送速率。实际上, PCE 可以使用远多于 3 个项, 例如 10 个或更多项。例如, 单一读取操作通常需要 6 个有意义的项来描述它: 1. 打开文件, 2. 开销, 3. 搜寻, 4. 传送, 5. 解压缩, 6. 关闭文件。其它操作(打开、关闭、解除链接、读取目录)典型地需要不同集合的项。

[0065] 作为对于线性方程的替换, PCE 可以扩展为一般多项式方程: 一个或多个项之和, 其中, 每一项是系数 (A) 乘以提升到某次幂的对应变量。例如, PCE 可以具有以下形式:

$$[0066] T_p = Ax + By + Cz^2 + Dw^{10} + Ev^{-2} \dots$$

[0067] 作为另一替换, PCE 可以扩展为任何普通方程, 其可以表示为一个或多个项之和, 其中, 每一项是系数 (A) 乘以对应变量的任意函数。例如, PCE 可以具有以下形式:

$$[0068] T_p = Ax + Blog y + Ce^z + D (w^2 + 2w + 1) + \dots$$

[0069] 作为简单示例, 线性 PCE 可以具有形式:  $T_p = Ax+By+Cz$ , 其中: x 是与请求 403 的开销有关的参数, y 是与每移动区段的头移动时间有关的参数, z 是与数据吞吐率有关的参数(例如用于读取或者写入指定量数据的时间)。系数 A、B 和 C 的值可以取决于调度队列 404

中请求 403 的位置。

[0070] 通过示例的方式,系数 A、B、C 可以根据状态以及请求 403 中的信息而得以确定。通过示例的方式,并且不失一般性,一条或多条信息可以包括设备的准备状态、以及所存取的最近逻辑块地址(LBA)。最近逻辑块地址可以指示驱动器的读取头的当前位置,因此允许计算搜寻距离(头行进距离)。

[0071] 在该示例中,系数 A 在此称为开销系数。该系数将以下事实纳入考虑:开销可以部分地取决于 I/O 请求 403 的属性以及介质设备 118 的特性。系数 B 可以与介质设备 118 中的读取头必须进行以便达到将要从其中读取数据的介质上的位置的头移动量有关。通过示例的方式,请求 403 可以调用介质设备 118 来读取始于区段 1000 的 10 个区段。在此情况下,数据系数 C 可以是待读取的区段之数量。移动系数 B 可以根据状态信息而得以确定,状态信息可以从设备介质层 306 来获得。状态信息可以包括读取头在 I/O 请求的起始处的初始位置。例如,如果读取头初始在区段 500 处,则移动系数 B 可以根据读取操作开始的区段(区段 1000)与当前区段(区段 500)之间的差而得以确定,例如  $B=1000-500=500$ 。调度器 302 可以在 I/O 请求 403 已经完成之后确定最终状态,例如,如果始于 1000 读取 10 个区段,则最终状态可以指示读取头结束于区段 1010。

[0072] PCE 也可以将用于填写 I/O 请求的不同可能情形纳入考虑。例如,如果介质设备 118 使用多层介质,则 PCE 可以将对从一个层到另一层切换然后移动读取头的预测时间  $T_p$  的效果纳入考虑。

[0073] 在某些实施例中,对于任何 I/O 请求所花费的实际时间  $T_a$  可以被测量并且与 PCE 系数一起使用从而对于未知变量的值迭代地求解 PCE。这样可以允许模型通过实际驱动器性能而得以恒定地更新。

[0074] 一旦计算出预测时间  $T_p$ ,调度器 302 就可以通过相似方式确定用于调度队列 406 中其它请求的预测时间。该过程可以按不同队列顺序而迭代地 重复,直到用于请求 403 的实际时间以及最佳调度队列顺序得以确定,如在 507 所指示的那样。注意,请求 403 的初始状态可以根据调度顺序中先前请求的结束状态而得以确定,而调度顺序中下一请求的初始状态可以根据请求 403 的最终状态而得以确定。

[0075] 调度器 302 可以预排调度队列中的各请求,对请求 403 的特性与已经在队列中的其它操作的特性进行比较。调度器 302 可以尝试队列中的每一可能位置处的请求 403、找寻优先级覆盖(priority override)、错过的最终期限、以及时序考虑。调度器 302 可以通过找寻所请求的 I/O 操作不错过其最终期限的顺序来确定最佳可能的新队列顺序。如果一个或多个操作必须错过它们的最终期限,则调度器可以使用不同的准则来确定队列顺序。可以影响调度队列 406 中的请求排序的额外准则可以包括(但不限于) I/O 请求优先级、流缓冲器状态、以及等待时间需求。在某些实施例中,用于一个  $T_p$  计算的 PCE 系数可以用于返回更新后的介质状态,其可以用作用于另一 PCE 计算的初始介质状态。

[0076] 根据一个示例,调度器 302 可以检查不同插入排列,例如将操作 4 插入在包含操作 1、2、3 的现有调度表中。每一排列可以产生不同的总时间 T。可能的排列和时间包括:

[0077] 4, 1, 2, 3 →  $T_1$

[0078] 1, 4, 2, 3 →  $T_2$

[0079] 1, 2, 4, 3 →  $T_3$

[0080] 1, 2, 3, 4 → T<sub>4</sub>

[0081] 调度器 302 可以确定哪种排列产生最小的预测总时间 T，并且使用该排列作为优选排列。

[0082] 此外，例如，通过合计用于每一调度项的时间，调度器 302 可以检查所预测的错过的最终期限并且查看这是否导致任何调度项错过其最终期限。这可以包括：确定每一调度操作的完成时间，并且对所述完成时间与该操作的最终期限进行比较。如果将错过任何最终期限，则可以将该操作重新调度到队列中更早的地方。

[0083] 在某些实施例中，例如，如果不能求解所有最终期限，则可以使用优先级。例如，如果将不可避免地错过某些最终期限，则最佳队列顺序可以是其中最低可能优先级操作错过它们的最终期限的队列顺序。如果存在适于前述考虑的多个可能队列顺序，则具有最少数量的同等优先级最终期限错过操作的队列顺序可以是最佳顺序。如果存在其中所有先前描述的考虑皆相同的多个可能队列顺序，则具有用于执行整个队列的最低总估计时间的队列顺序可以是最佳顺序。在某些情况下，只要高优先级操作可以满足其最终期限，就可以在高优先级操作之前调度低优先级操作。

[0084] 如果存在其中所有先前描述的考虑皆相同的多个可能队列顺序，则其中调度队列 406 中的最新操作去往队列末尾的顺序可以是最佳顺序。如果调度器 302 必须估计用于将新操作插入队列中的总队列执行时间，则其可以考虑多种不同因素。例如，调度器可以考虑调度器 302 恰在队列顺序估计之前检索的设备驱动器的当前状态。此外，调度器 302 可以考虑 FIOS 堆栈 300 中介质过滤器层和其它层所报告的性能系数。这些值可以包括例如头移动、头对准、DMA 设置、数据传送率、对于存储层的层改变等。这些值可以通过 I/O 性能的运行时间测量而得以馈送，从而估计可以是尽可能真实的。

[0085] 一旦调度器 302 已经对于请求 403 的操作确定队列 406 中的最佳位置，该操作就可以插入在此，如图 4 中的 407 和图 5 中的 508 所指示的那样。再次参照图 4，如果存在可用于执行 I/O 请求的资源，则调度器 302 可以将调度队列 406 中的第一操作移动到发出队列 408，如在 409 所指示的那样。操作的关联介质请求可以是从发出队列 408 执行的。为了执行介质请求，调度器 302 可以将请求传递下达 FIOS 堆栈 300 中的第一层。

[0086] 调度器 302 之下的每一层可以通过之上的层看到传递至其的介质请求 403。如果适当，则层可以处理数据。例如，如果请求是读取操作，则解存档器层 301 可以针对开放存档的内容检查所提供的路径名称，如果其找到文件，则可以将请求重新映射为读取压缩后的数据。FIOS 堆栈 300 中的每一层将处理过的或未处理的介质请求传递给下一更低层，直到请求最终到达硬件层 310。当硬件层 310 对请求进行响应时，该响应从下至上遍历堆栈 300 中的每一层，并且如果适当，则每一层可以处理检索到的数据。例如，解存档器层 301 可以获知所返回的数据必须被解压缩，从而其对它进行解压缩，之后将响应传递回至堆栈。该响应最终回到调度器 302，得以完成。

[0087] 当检索到的数据回到堆栈时，调度器 302 可以接收它，并且然后将 I/O 操作移动到完成的队列，其可以触发对于应用 401 的回调函数 411（如果应用设置了回调）。替换地，应用 401 可以轮询 FIOS 堆栈 300，以确定是否已经完成所请求的 I/O 操作。一旦已经完成 I/O 操作，其将可以移动到释放操作池 412。释放操作池 412 可以包含未使用的一组 I/O 操作结构。这些操作可以包括已经分配给客户机应用的操作或已经为客户机应用所使用并且然

后再次为了使用而释放的操作。当客户机应用进行 I/O 请求时,调度器 302 可以从该池将 I/O 操作分配给客户机。释放 I/O 池 412 可以实现为堆栈。释放 I/O 请求可以从释放操作池 412 弹出,并且推送到输入队列 404。以此方式,释放 I/O 请求可以被重新使用。

[0088] 根据本发明实施例,调度器 302 可以通过调度循环而操作,如下:

[0089] 1. 检查 I/O 完成

[0090] 2. 发出新 I/O 请求

[0091] 3. 发出 I/O 回调(若有的话)

[0092] 4. 插入到调度表(从输入调度插入)

[0093] 5. 再次新发出

[0094] 6. 检查预测出的错过的最终期限

[0095] 7. 返回到 1。

[0096] 对于循环的每一迭代的调度插入的数量可以受限于某个最大数,例如十六次插入。

[0097] 在本发明某些实施例中,辅助处理器单元 105B (例如 SPE 206)可以通过关于主处理器 105A 或 PPE 204 为原子的交换而将对 I/O 的请求添加到输入队列 404 来请求其自身的 I/O。例如,在传统单元处理器实现方式中,SPE 206 可能没有任何 I/O 设施。然而,如果输入队列 404 是公共数据元素,则辅助处理器可以通过与主处理器 105A 的标准原子交换并且发信号通知主处理器 105A 来将 I/O 请求添加到队列。

[0098] 在很多现有技术实现方式中,如果辅助处理器需要用于即刻处理的数据,则该数据必须在主存储器中。通过本发明实施例,通过对照,辅助处理器 105B 可以将 FIOS 堆栈 300 触发为去从硬介质设备 118 或者甚至在网络 127 上得到所需的数据。

[0099] 通过示例的方式,而非通过限制的方式,输入队列 404、完成队列 410 以及释放池 412 可以是原子堆栈。辅助处理器 105B 可以弹出来自释放池 412 的操作,填充用于该操作的路径和偏移,将操作推送回到输入队列 406,然后执行同步,并且唤醒调度器 302。辅助处理器可以在为了完成 I/O 操作而进行中断式轮询的同时进行其它工作。

[0100] 通过示例的方式,在单元处理器系统 200 中,SPE 206 可以使用以下类型的指令的序列来服务于 I/O 请求:

[0101] Op = pop(FREE)

[0102] Op path =.....

[0103] Offset =.....

[0104] Push (op, incoming)。

[0105] SPE 206 所请求的数据可以发送到可由 PPE 204 寻址的任何地方。如果 SPE 206 被锁定,则 FIOS 300 可以将数据直接写入到 SPE 206 的本地存储 LS。在某些实施例中,一个 SPE 206 可以请求解存档器进行与另一 SPE 的解压缩。可以例如使用各处理器之间的远程过程调用(RPC)来实现该操作。在此示例中,SPE 206 要求 PPE 204 为其做事。在更传统的 RPC 中,这是大约另一种方式。

[0106] 预取

[0107] 在某些实施例中,调度器缓存 305 (例如 HDD 缓存)可以用于将文件从介质设备预取到存储设备 115 上的缓存 117,从而该文件可以稍后得以快速读取。在此情况下,预取操

作 413 可以直接插入到发出队列 402。预取可以是很多类型的缓存已经实现了的标准手动预取，例如在 PowerPC CPU 架构中的“dcbt”指令。根据本发明实施例，预取可以排队并且作为上述调度器循环的一部分而得以执行。预取可以在其它 I/O 请求得以履行的同时通过相对高的等待时间以及必须存取的低吞吐量来促进与相对慢的源介质（例如蓝光和 UMD）的工作。

[0108] 调度器 302 可以被配置为以相对低的优先级实现这种预取，从而其将仅运行在介质设备 118 将另外为空闲的未用时刻，并且将不干扰到其它 I/O 请求。例如，系统 300 可以具有慢介质设备 118（例如，比如蓝光盘（BD）驱动器的光盘）和较快存储设备 115（例如硬盘驱动器）。调度器缓存层 305 可以用于将文件从光盘驱动器异步拷贝到硬盘。当文件稍后被存取时，其将以更高的 HDD 速度（例如 20MiB/s）而不是更慢的光盘速度（例如 8MiB/s）而得以读取。

[0109] 虽然缓存预取通常是对于硬盘缓存而完成的，但预取也可以对于主存储器 106、206 而完成。预取操作可以按在调度完成之后它们被接收的顺序而被包括在调度队列 406 的结束。在某些实施例中，预取操作可以根据需要而延迟。在某些实施例中，调度器 302 所使用的性能特征方程可以包括关于什么存储于缓存 117 中的信息，作为状态信息的一部分。预取操作并非是以与其它 I/O 操作相同的方式而调度的。预取操作保存在分离的预取队列 413 中，预取队列 413 仅当调度队列 406 为空时受服务。预取操作可以根据需要而延迟。另外，预取可以是按它们被接收到的顺序而执行的。调度器 302 可以保持预取请求排队，并且仅当 I/O 子系统已经空闲指定的时间长度时执行它们。这样防止预取干扰到正常 I/O 请求。此外，预取也不限于单一缓存块；它们可以是任何大小的，或者甚至传递告诉缓存从开始到结束加载整个文件的特殊“完整文件”值。进而，虽然预取可以是任何大小的，但预取可以被实现使得不多于一个的缓存块被填充，之后返回到调度器 302，以检查 FIOS 堆栈 302 继续空闲。

[0110] FIOS 堆栈 300 可以被配置为以避免不必要的预取的方式来实现预取。例如，如图 6 所示，如果缓存 601 中没有块，则 FIOS 调度器缓存层 305 可以填充第一块，然后返回以检查空闲状态。如果块 1 至 N 在存储设备 115 上的缓存 117 中，则第 (N+1) 块将填充有来自介质设备 118 的数据。调度器缓存层 305 可以返回，以检查空闲状态。该过程可以重复，直到新 I/O 到达输入队列。这些特征允许任意数量的预取排队，从而对于应用而快速且高效地将大量数据从介质设备 118 预取到存储设备 115（或主存储器 106），而不干扰到正常调度的 I/O 请求。

[0111] 例如，在游戏应用中，游戏数据一般存储在由介质设备 118 所读取的介质（例如 CD-ROM 或 HD-DVD 盘）上。游戏可以确定玩家正移动朝向特定目的地，并且计算出玩家应在 60 秒内到达那里。在该时间期间，FIOS 堆栈 300 可以从介质设备预取与目的地有关的文件。如果玩家决定转向并且前往不同的目的地，则应用可以取消预取。在其它实施例中，应用可以使用高级通知来防止覆盖缓存 117 中的所需数据。

[0112] 如上所述的缓存预取可以改进以 I/O 驱动的应用（例如视频游戏）以及具有特定 I/O 需求的平台（例如新潮视频游戏控制台）方面的 I/O 性能。具体地说，游戏数据通常存储在慢介质（例如光学介质或网络文件服务器）上，但游戏可以具有对快速本地存储（比如 HDD）的存取。

[0113] 利用辅助处理器的数据变换

[0114] 根据另一实施例，FIOS 利用特定处理器架构，所述架构利用主处理器和辅助处理器，所述辅助处理器具有关联本地存储器。这些实施例可以例如用于数据变换，例如解压缩或解密。通过比较的方式，特定先前解存档系统已经使用标准背景线程中运行的简单解压缩算法(例如 zlib)。通过对照，本发明实施例可以包括抽象编解码器接口，其允许标准解压缩器实现方式和非标准实现方式被配置为利用包括带有关联本地存储器的辅助处理器的处理器架构(例如单元处理器架构)。这些实施例允许解压缩器对象设置多个标志和约束，解存档器 301 于是可以利用其来优化存储器使用和速度。这些约束中的很多以及导致特殊处理的情形对于游戏编程、文件系统 I/O 解压缩、或系统 100 的架构的情形是唯一的。

[0115] 特定应用(例如视频游戏应用)具有受限量的可用存储器，这使得期望使用尽可能少的存储器。一个通常表述是：对于 I/O 解压缩，仅具有三个 I/O 缓冲器：由读取占据的一个缓冲器，由数据变换操作(例如解压缩或解密)占据的多达两个临时缓冲器(一个用于输入，一个用于输出)。通过示例的方式，而非限制，特定实施例可以通过对于数据变换使用带有关联本地存储器(例如单元处理器 200 中的 SPE 206)的辅助处理器而完全避免使用临时缓冲器。

[0116] 例如，如果解压缩是由主处理器(例如 PPE 204)处理的，则典型地需要一个缓冲器来临时存储压缩后的数据，并且需要另一缓冲器来临时存储解压缩后的数据。如果带有本地存储器 106B 的辅助处理器 105B 可用，则可以使用本地存储器 106B 而非临时缓冲器来完成相同的解压缩。压缩后的数据与解压缩后的数据皆可以存储于本地存储器 106B 中。通过示例的方式，在特定单元处理器架构中，SPE 206 具有 256K 字节的本地存储容量。这对于压缩后的输入、解压缩后的输出以及用于解压缩算法的代码通常是足够的存储器空间。通过比较，如果解压缩是由主处理器 105A(例如 PPE 204)处理的，则压缩后的数据读取到第一缓冲器，解压缩，并且解压缩后的数据写入到第二缓冲器。解压缩后的数据于是从第二缓冲器拷贝到其目的地。

[0117] 图 7A 和图 7B 示出其中利用辅助处理器实现特定数据变换的两个示例。具体地说，如图 7A 所示，在履行 I/O 请求的过程期间，主处理器 105A 上运行的 FIOS 300 可以例如经由数据变换层 311 调取数据变换。主处理器 105A 可以设置辅助处理器 105B，如在 702 所指示的那样。具体地说，主处理器 105A 可以命令辅助处理器 105B 加载编码后的指令 701，用于将数据变换置入到辅助处理器的本地存储器 106B。主处理器或辅助处理器于是可以通过 FIOS 堆栈 300 工作，以将数据 703 从介质设备 118 传送到主存储器 106，如在 704 所指示的那样。辅助处理器 105B 于是可以将数据 703 加载到本地存储器 106B，如在 706 所指示的那样。

[0118] 辅助处理器于是可以对数据 703 实现变换，如在 708 所指示的那样，以产生变换后的数据 705，变换后的数据 705 可以存储于本地存储器 106B 中。通过示例的方式，编码后的指令 701 的执行可以对数据 703 进行解密或解压缩。变换后的数据 705 可以传送到主存储器 106，如在 710 所指示的那样。替换地，变换后的数据 705 可以传递到某个另外目的地，例如存储设备 115。前述操作序列可以对于后续数据块而重复，直到已经处理完 I/O 请求所指定的所有数据。注意，一旦输入数据 703 已经传送到本地存储器 106B，就可以通过新输入数据或变换后的数据 705 在主存储器中对其进行覆盖。

[0119] 通过示例的方式,而非通过限制的方式,单元处理器架构(例如图 2 中所描述的架构)可以用于实现根据图 7A 的数据解压缩。与文件系统 I/O 解压缩的需求相结合的 SPU 编程的特性(其中,数据被流送到 SPE 206 的本地存储 LS,然后一旦处理完成就流送回去)为改进的 I/O 性能提供了机会。例如,使用 SPE 206 来解压缩可以提供从 37Mbps 到 42Mbps 的性能改进,结果是释放出额外的缓冲器。可以通过 Sony Playstation 2 中所使用的处理器架构来实现相似的解压缩。在此情况下,主处理器称为情绪引擎(EE),各辅助处理器之一(称为 I/O 处理器(IOP))具有关联本地存储器。在此情况下,沿着图 7A 所示的线,解存档器 301 可以运行在 EE 上,解压缩可以实现在 IOP 上。替换地,图 7A 所描述的技术可以用于分布式计算架构,其包括两个带有分离可寻址存储器的处理器。

[0120] 注意,虽然前述示例处理通过变换层所实现的数据变换,但关于图 7A 而描述的方法可以用于实现任何介质过滤器层 304。通过对照,现有技术文件 I/O 系统典型地通过主处理器(例如单元处理器示例中的 PPU)而实现。

[0121] 在替换实现方式中,如果从介质设备 118 读取的数据之目的地是慢存储器(例如图形存储器 140,比如视频随机存取存储器(VRAM)),并且编解码器需要执行对其输出的随机存取,则 FIOS 解存档器可能需要分配临时输出缓冲器。然而,如果编解码器不需要随机存取(正如例如以 SPU 实现的解压缩器的情况),则 FIOS 解存档器 301 可以避免分配临时输出缓冲器,并且数据 703 可以直接解压缩到目的地。进而,文件系统解压缩可以要求“溢出”或未用的字节;即,当恰需要一些字节时对整个 64 千字节块进行解压缩。如果编解码器指示其可以输出恰一部分压缩后的块(正如例如以 SPU 实现的解压缩器的情况),则 FIOS 解存档器可以避免分配临时输出缓冲器。此外,带有关联本地存储器 106B 的辅助处理器 105B 可以用在上述各情形的任何以及所有组合中。

[0122] 图 7B 示出结合慢图形存储器 340 使用辅助处理器 105B 和本地存储器 106B 的示例。FIOS 300 可以将数据从介质设备 118 读取到图形存储器 340(例如 VRAM)。使用缓冲器将数据传送到 VRAM 势必缓慢。然而,将 VRAM 数据读取到特定类型的辅助处理器本地存储器(例如 SPE 206 的本地存储 LS)不成问题。在图 7B 描述的示例中,FIOS 300 主要是在主处理器 105A 上实现的。未被变换的数据 703 可以从图形存储器 340 传送到与辅助处理器 105B 关联的本地存储器 106B。FIOS 301 可以将变换指令 701 加载到本地存储器 106B,以由辅助处理器 105B 执行。辅助处理器 105B 可以通过执行指令 701 而将数据 703 变换为变换后的数据,以生成变换后的数据 705。变换后的数据于是可以传送到图形存储器 340。在对该实施例的某些变形中,未被变换的(例如加密的或压缩后的)数据 703 可以是从介质设备 118 读取的,并且存储于主存储器 106 中的缓冲器中,之后其传送到本地存储器 106B。

### [0123] 嵌套式存档

[0124] 先前 FIOS 实现方式已经包括了随机存取解存档系统,其允许压缩后的存档的内容实际上出现于文件系统等级中,并且其通过覆盖读取和解压缩而以优化高效的方式从这些存档检索数据。本发明实施例通过如下方式细化该系统,即:包括用于处理具有单一解存档器层 301 的嵌套式存档的能力,并且没有性能损失(penalty)。在此所使用的术语嵌套式存档指代存储于其它存档内的存档,其自身可以存储于另一存档中,到达嵌套的任意深度级别。

[0125] 对于满足游戏 I/O 和数据布局的唯一需求而言,嵌套式存档支持是特别有用的。

使用嵌套式存档可以与调度器 302 所使用的驱动器模型 302A 协作。具体地说，大型文件允许更好的模型驱动器头移动。如果压缩后的存档格式为随机存取，则嵌套式存档支持可以大受促进。通过对照，典型地，压缩后的存档并非随机存取。例如，zip、gzip、7z、压缩格式等典型地要求从存档的开头线性存取。

[0126] 在此所使用的术语“存档”指代组合为单个较大文件的一组数据文件。源数据可以不修改地被存储，或者一个或多个数据变换（例如压缩或加密）可以应用于数据的任何部分。索引（有时称为内容表）可以通过描述源文件的至少某方面（例如存档内的路径和偏移）的存档而得以保存，从而它们可以被查找并且单独存取。术语“嵌套式存档”指代其中一个或多个源数据文件为另一存档（或嵌套式存档）的存档。根据这种嵌套，任何给定源数据块可以具有应用至其的零个或多个数据变换。例如，具有两个级别（级别 1 和级别 2）的视频游戏应用可以使用题为“game.psarc”的嵌套式存档，其包含题为“level1.psarc”和“level2.psarc”的两个存档。这可以为了除错以及测试的目的而通过允许游戏开发者使用单存档（“level1.psarc”、“level2.psarc”等）来促进开发，而非总是每次花费时间来创建更大的“game.psarc”存档。

[0127] 开发者可以生成嵌套式存档，但传统 I/O 读取器无法读取或者无法高效地读取多级存档。为了克服该问题，存档可以如图 8 所示而被配置。具体地说，外部存档 800 可以包括内部存档 801、802，其包含带有用于单独级别游戏的数据的文件。通过示例的方式，而非通过限制的方式，内部存档 801、802 可以包含与特定游戏级别有关的数据，例如地图、在该级别可用的武器以及在该级别遭遇的敌人。用于内部存档 801、802 的数据可以被单独压缩，并且与公共数据 803 绑定。外部存档 800 内的文件可以在一块接一块的基础上被压缩。每一存档可以包括允许解存档器在外部存档 800 内的任何地方（包括内部存档 801、802 中的任何地方）找寻任何给定数据块的索引。该索引可以被看作用于存档的内容表的等同物。

[0128] 当 FIOS 300 发出对来自存档特定文件的请求时，该请求可以看起来如此：Request:/game/level2/map(1000, 10)。部分“/game/level2/map”称为路径，其用于提交特定文件。部分“(1000, 10)”标识请求的偏移和长度。当将要读取外部存档 800 或其一部分时，解存档器 301 可以考虑可用文件的列表。该列表可以根据哪些存档打开而不同。例如，如果外部存档 800 或内部存档 801、802 皆未打开，则列表可以看起来如此：

[0129] /game.psarc

[0130] 如果外部存档 800 打开，则列表可以看起来如此：

[0131] /game (folder)

[0132] /game/level1.psarc

[0133] /game/level2.psarc

[0134] /game/common.dat

[0135] 如果内部存档打开，则列表可以看起来如下：

[0136] /game/level1 /map

[0137] /game/level1/weapons

[0138] /game/level1 /enemies

[0139] /game/level2/map

[0140] /game/level2/weapons

[0141] /game/level2/enemies

[0142] /game/common.dat

[0143] 每一存档 800、801、802 可以具有内容的路径(或部分路径)的散列的列表。散列列表可以用于生成用于存档的内容表(TOC)。例如,用于外部存档 800 的内容表可以被配置如下:

[0144] TOC:hash (/level1.psarc)

[0145] hash (/level2.psarc)

[0146] hash (/common.dat)

[0147] 其中,hash() 指代圆括号中的量的散列化。可以使用任何合适的散列,例如 MD5 散列。

[0148] 相似地,用于级别 1 内部存档 801 的内容表可以被配置如下:

[0149] TOC:hash (/map)

[0150] hash (/weapons)

[0151] hash (/enemies)

[0152] 如果使用图 8 所示的类型的嵌套式存档,则解存档器 311 可以被配置为处理它们。通过示例的方式,解存档器 311 可以实现图 9 所示的解存档方法 900。方法 900 可以开始于存档中的初始查找,如在 902 所指示的那样。通过示例的方式,而非通过限制的方式,初始查找可以返回用于搜寻的数据的初始路径、偏移、长度、以及所需数据变换的排序后的列表。具体地说, I/O 请求可能是如下的:

[0153] (/game/level2/weapons, offset 0, length 50000, none)

[0154] 内部存档中的该文件的初始查找可以返回以下内容:

[0155] (/game/level2.psarc, offset 20000, length 30000, zlib decompress)

[0156] 在初始查找之后,解存档器 311 可以求解嵌套,如在 904 所指示的那样。在求解嵌套中,解存档器 311 可以尝试确定存档器 802 是否存储于另一存档内以及其是压缩过还是未压缩而存储的。

[0157] 在求解嵌套中,解存档器 311 可以返回用于搜寻的数据的路径、偏移、长度、以及所需数据变换的排序列表。通过示例的方式,路径可以具有形式:/game.psarc, offset 120000, length 30016, decrypt + zlib decompress。嵌套可以通过循环而不是递归而求解,以避免堆栈极限。一旦已经求解嵌套,就可以从介质设备 118 传送数据。通过示例的方式,而非通过限制的方式,解存档器 311 可以一块接一块地迭代地读取并且变换数据,如在 905 所指示的那样。具体地说,解存档器可以读取数据块,如在 906 所指示的那样,然后将数据变换应用于所读取的块,如在 908 所指示的那样。多级嵌套存档可以被求解,从最外存档下至单一交错读取和变换循环读取块,以最高效地使用源介质。在已经读取并且变换了每一块之后,可以将期望的部分拷贝去到合适的目的地,例如在主存储器 106、存储设备 115 或其它地方,如在 910 所指示的那样。块的期望部分可以是任何大小,例如与单个字节同样小,或者与整个块同样大。

[0158] 覆盖层

[0159] 如上所述,FIOS 堆栈 300 中的介质过滤器层 304 可以包括覆盖层 307。根据本发明实施例,覆盖层 307 可以被配置为允许在文件系统级别任意覆盖文件和目录。普通构思

可以与 Unix 中的联合安装(union mount)比较,例外在于其远更灵活。典型的联合安装通过将一个文件系统中的目录与另一文件系统的整个内容合并而工作。覆盖层 307 进行相似的事情,但远更灵活并且全特征化。例如,覆盖层 307 可以在目录、文件或甚至文件中的字节范围的粒度上操作。目录可以被隐藏,或者使用覆盖层 307 来代替它们的内容。此外,覆盖层 307 可以用于将从主要数据源所请求的文件替代为来自次要数据源的其它文件。次要数据源可以是文件或目录,其可以在与主要数据源或另一介质相同的介质上。此外,第二数据源可以甚至是虚拟数据源,从而数据可以通过回调而得以指定。在这种特定情境下,术语“回调”指代作为对于其它代码的变元而传送的可执行代码。

[0160] 进而,创建覆盖的应用可以指定准则,用于覆盖层 307 应该如何以及何时应用覆盖。例如,如果应用 103 正将目录 B 覆盖到目录 A,则所述准则可以是“覆盖”=首先查找 B 中的文件,然后查找 A 中的文件;“未覆盖”=首先查找 A 中的文件,然后查找 B 中的文件;“较新的”=如果文件存在于这两个地方,则使用带有更近修改日期的那个;“较旧的”=如果文件存在于这两个地方,则使用带有较早修改日期的那个;诸如此类。

[0161] 覆盖层 307 可以被配置为允许将针对来自特定源的 I/O 读取的请求任意映射到来自另一源的读取。例如,对于从介质设备 118 读取特定文件的请求可以映射到位于例如存储设备 115 上的对应文件。使用这种范围从文件内的具体字节到整个目录的特征数据可以被隐藏并且四处交换。通过指定哪些目录、文件或字节需要待替代,覆盖层 307 可以使得 FIOS 堆栈 300 必须进行以处理特定 I/O 请求的工作量最小化。

[0162] 覆盖层 307 可以用于例如实现对软件和 / 或数据的更新。在很多现有技术更新系统中,可执行的可链接文件(ELF)从介质设备 118 拷贝到存储设备 115,然后利用更新后的代码或数据打补丁。通过本发明实施例,通过对照,应用 103 可以命令 FIOS 101 填充带有来自介质设备 118 的数据的特定字节范围以及来自某个另外位置(例如存储设备 115)的其它字节范围。因此,不是必须拷贝整个文件来对其打补丁。反之,打补丁操作可以实现在 I/O 系统级别。

[0163] 多数现有打补丁实现方式是基于替换整个文件的。这是最便于实现的,并且因此至今是最常用的方法。通过对照,覆盖层 307 可以实现的增强型打补丁操作允许远更复杂且有用的模式,例如在压缩后的并且加密后的存档内打补丁。

#### [0164] 速度仿真层

[0165] 在本发明特定实施例中,速度仿真层 315 可以用于在开发期间减慢较快存储设备,以仿真较慢 I/O 设备的性能。速度仿真层 315 可以利用搜寻时间、吞吐量以及较慢 I/O 设备的其它性能特征的模型。例如,速度仿真层 315 可以减慢存储设备 115(例如硬盘驱动器)来仿真介质设备 118(例如 HD-DVD,比如蓝光驱动器)的较慢搜寻时间和 / 或吞吐量。

[0166] 根据本发明实施例,速度仿真层 315 可以通过将先前会话期间收集到的驱动器性能数据馈送到后来会话而得以实现。如上所述,调度器 302 所使用的性能模型可以被配置为对任意复杂驱动器特性(比如吞吐量、激光器摇摆、读取头移动、层改变、以及甚至请求开销)进行建模。速度仿真层 315 可以使用该信息提供特定介质设备的高精度仿真。由于这种仿真是基于运行时间性能的,因此其将远比在忽略更复杂细节(比如搜寻时间)的同时尝试简单地匹配吞吐量的简单质朴仿真层更精确。

[0167] 本发明实施例提供了在利用大量 I/O 的应用和系统中的改进的 I/O 性能。如上所

述,本发明实施例在视频游戏应用和视频游戏系统中尤其有用。然而,本发明实施例不限于这些应用和系统。

[0168] 虽然以上是本发明优选实施例的完整描述,但有可能使用各种替换、修改和等同物。因此,本发明范围不应参照以上描述而得以确定,而应反之参照所附权利要求连同其等同物的全部范围而得以确定。在此描述的任何特征,无论优选与否,可以与在此描述的任何另外特征组合,也无论优选与否。在所附权利要求中,不定冠词“一”或“一个”指代不定冠词后的一个或多个项目的数量,除非另外明确声明。所附权利要求不应理解为包括装置加功能的限定,除非这种限定使用短语“用于……的装置”而明确陈述于给定权利要求中。

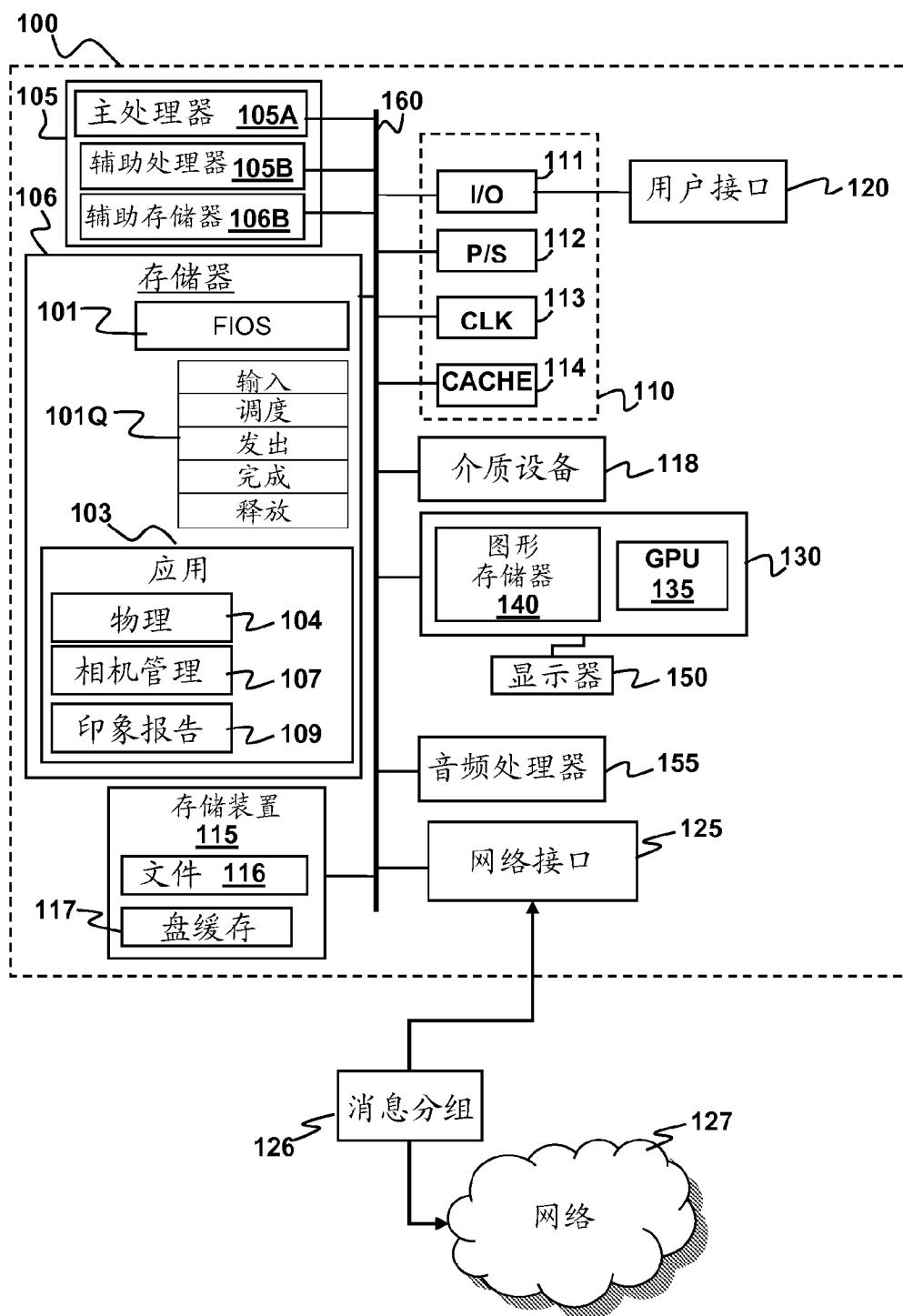


图 1

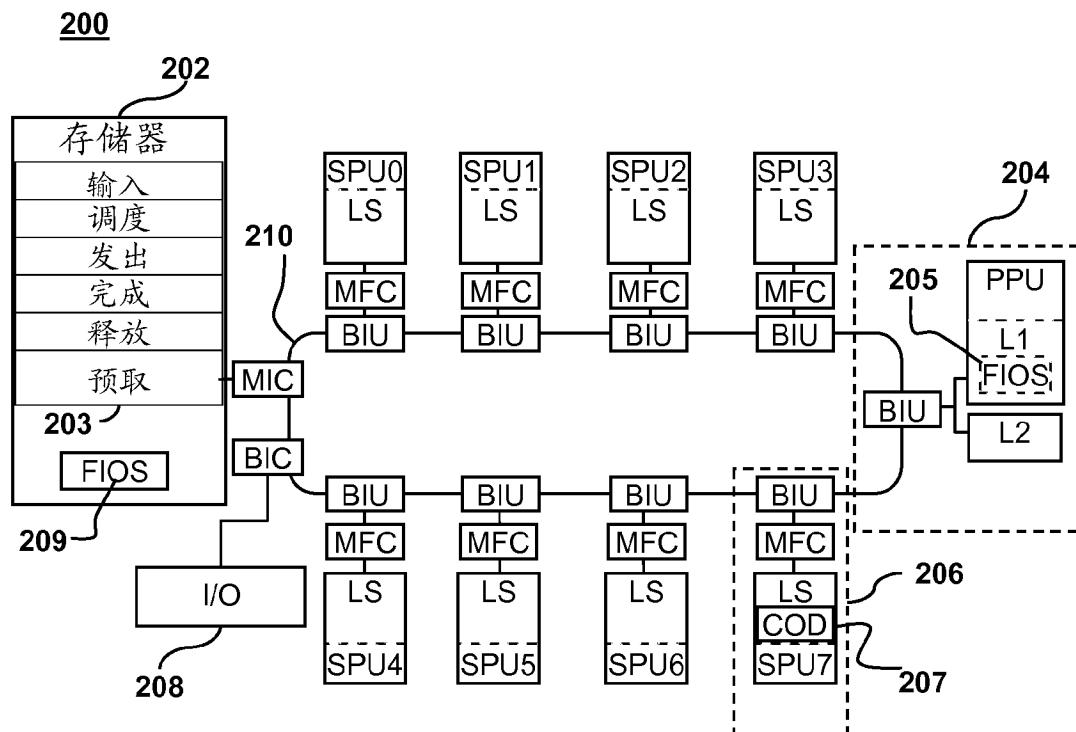


图 2

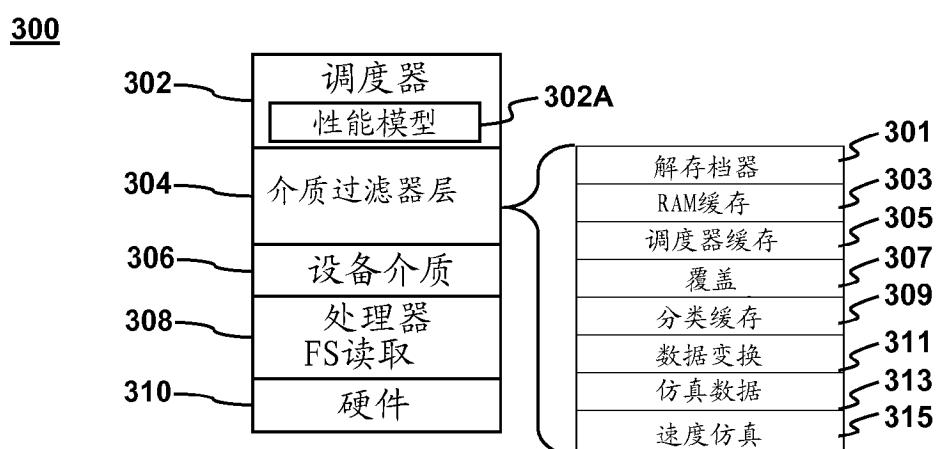


图 3

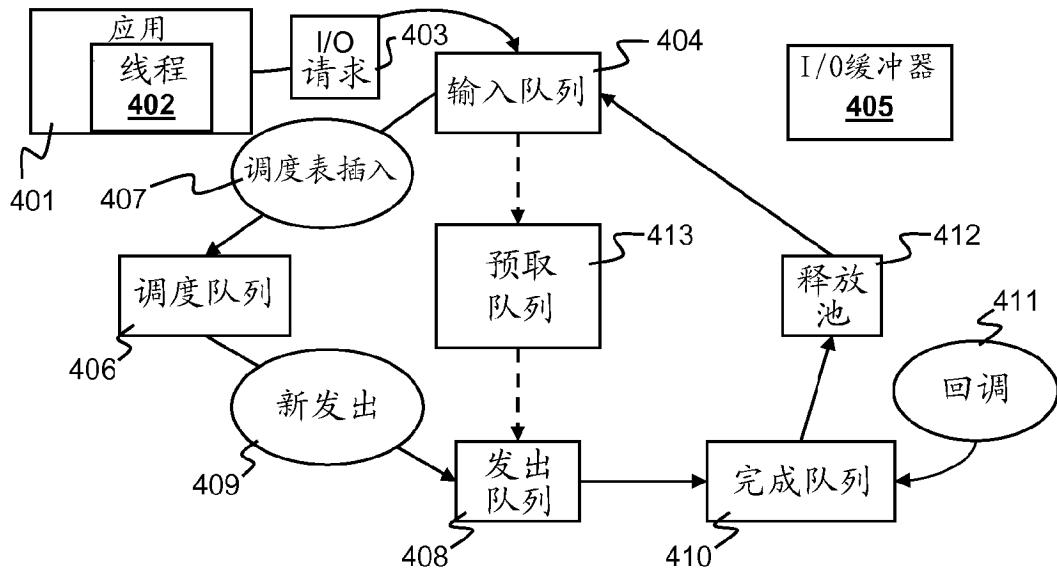


图 4

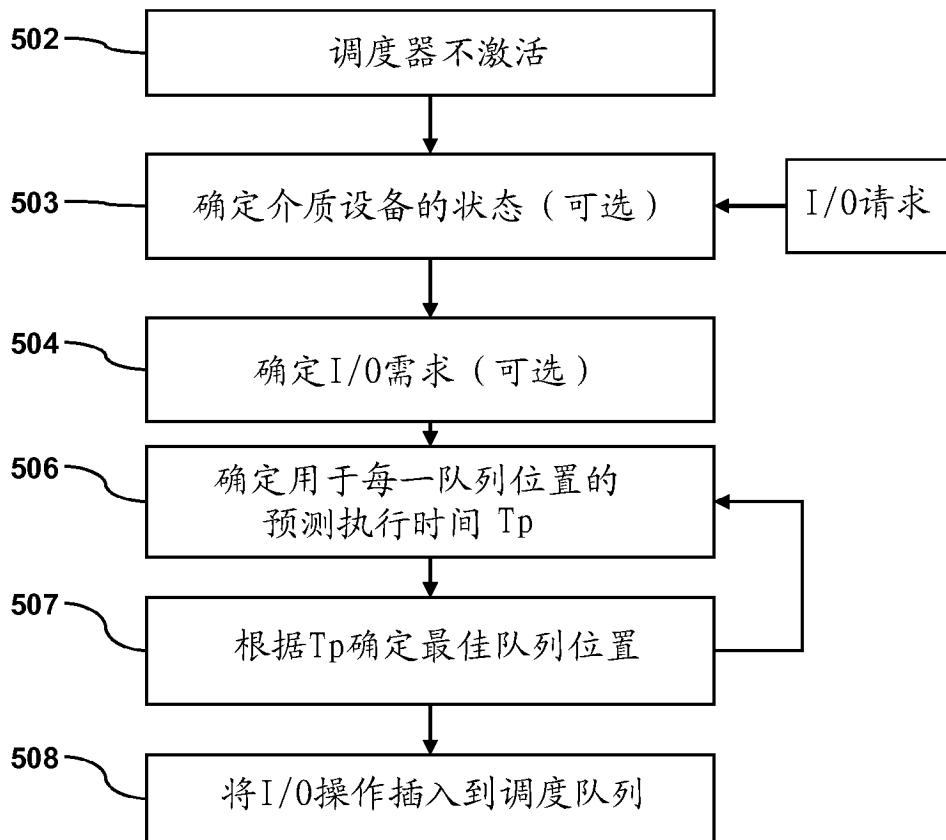
**500**

图 5

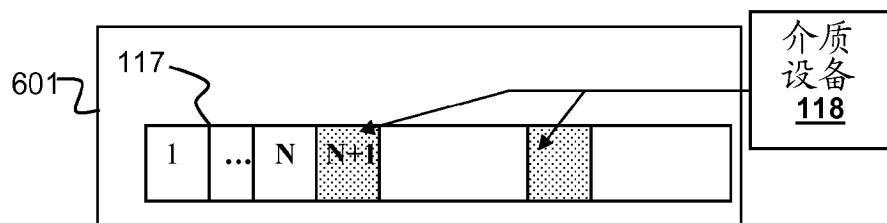


图 6

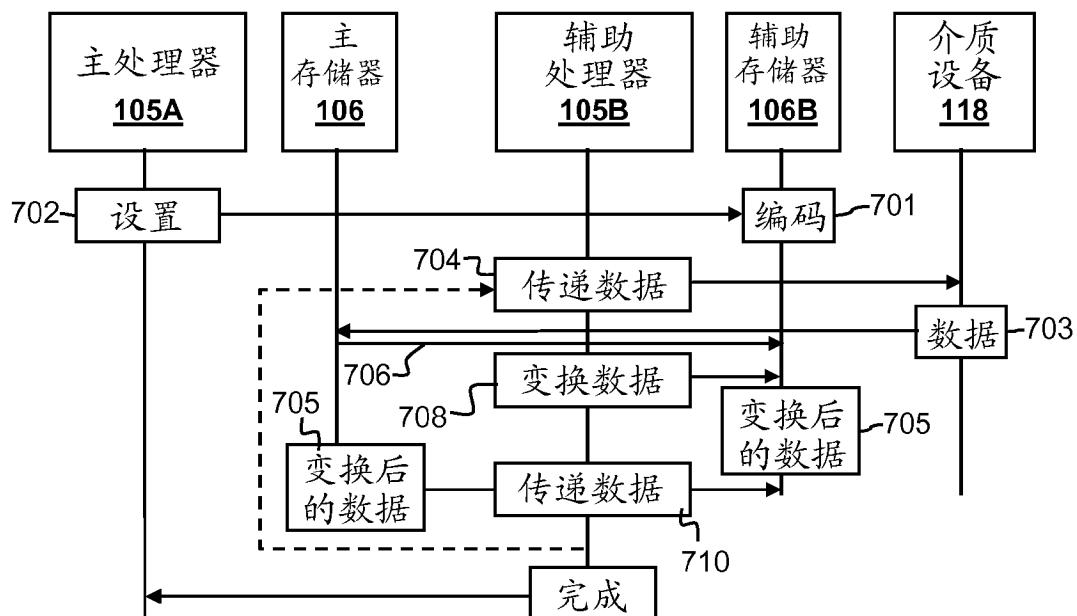


图 7A

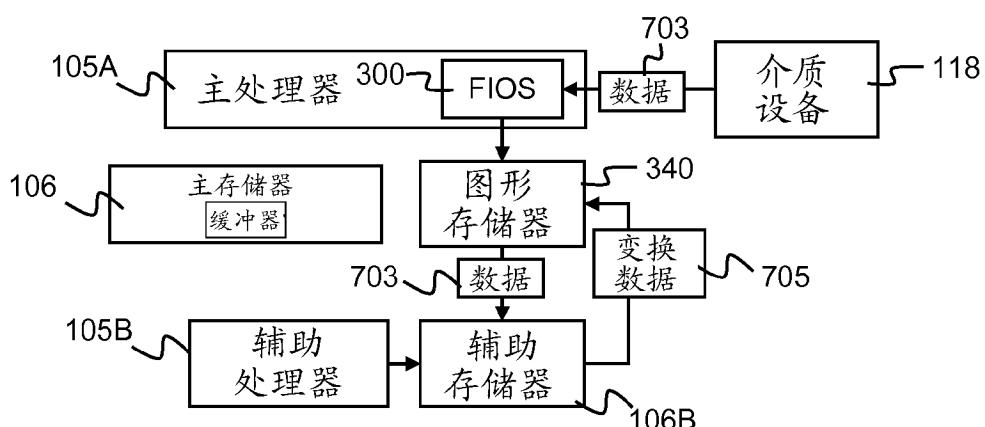


图 7B

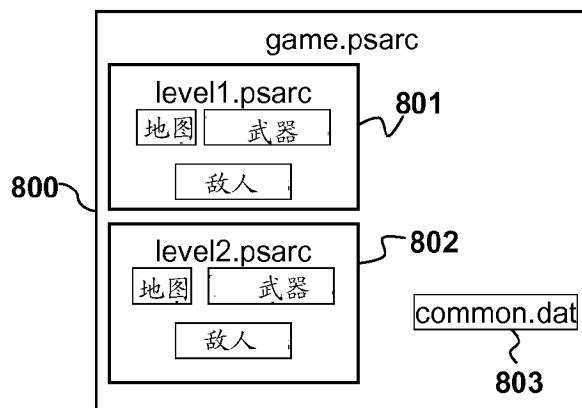


图 8

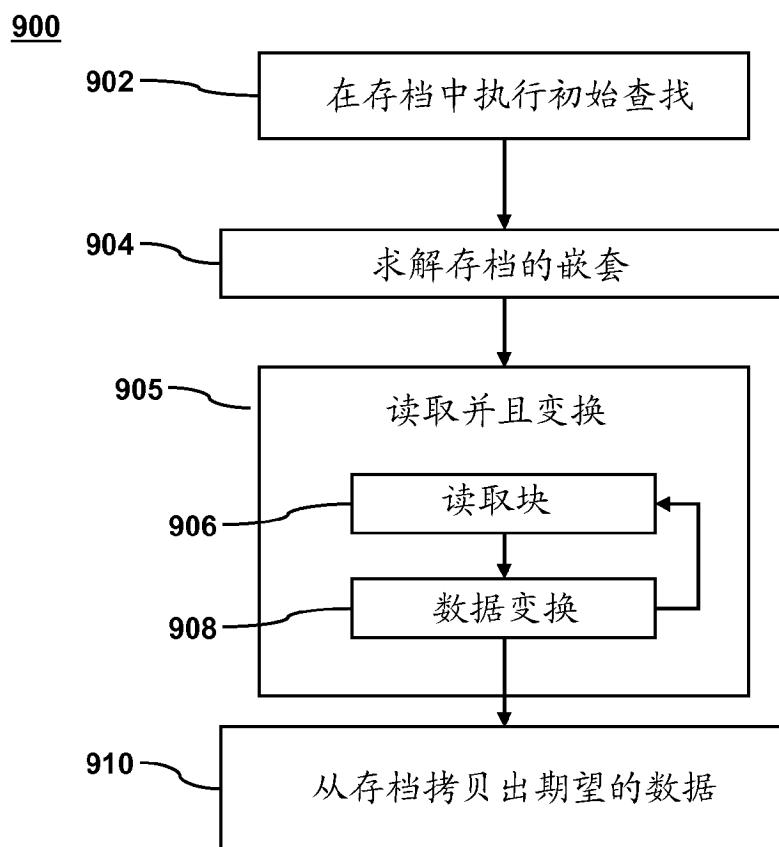


图 9