(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国 际 局

(43) 国际公布日
2016 年 8 月 4 日 (04.08.2016)

WIPO | PCT

(10) 国际公布号
WO 2016/119605 A1

(51) 国际专利分类号:
G06F 9/445 (2006.01)

(21) 国际申请号: PCT/CN2016/071165

(22) 国际申请日: 2016 年 1 月 18 日 (18.01.2016)

(25) 申请语言: 中文

(26) 公布语言: 中文

(30) 优先权:
201510042012.5 2015 年 1 月 27 日 (27.01.2015) CN

(71) 申请人: 阿里巴巴集团控股有限公司 (ALIBABA GROUP HOLDING LIMITED) [GB/CN]; 英国开曼群岛大开曼乔治城资本大厦一座四层 847 号邮箱, Cayman Islands (GB)。

(72) 发明人; 及

(71) 申请人 (仅对美国): 朱光宇 (ZHU, Guangyu) [CN/CN]; 中国浙江省杭州市余杭区文一西路 969 号 3 号楼 5 楼阿里巴巴集团法务部, Zhejiang 311121 (CN)。

(74) 代理人: 北京国昊天诚知识产权代理有限公司 (CO-HORIZON INTELLECTUAL PROPERTY INC.); 中国北京市朝阳区西坝河西里 28 号英特公寓 C 座 104, Beijing 100028 (CN)。

(81) 指定国 (除另有指明，要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW。

(84) 指定国 (除另有指明，要求每一种可提供的地区保护): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO,

[见续页]

(54) Title: METHOD FOR MEMORY ALLOCATION DURING APPLICATION START IN TERMINAL AND SYSTEM THEREOF

(54) 发明名称 : 终端中应用启动时的内存分配方法及其系统

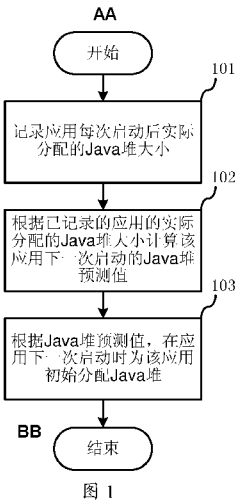


图 1

101 The size of a Java heap actually allocated after every start of an application is recorded
102 On the basis of the recorded size of the actually-allocated Java heap, a predicted value of the Java heap at the time of next start of the application is computed
103 Initial allocation of the Java heap for the application is performed on the basis of the predicted value of the Java heap when the application is started next time
AA Begin
BB End

(57) Abstract: Disclosed are a method for memory allocation during application start in a terminal and a system thereof, relating to terminal application technologies. On the basis of the recorded size of a Java heap actually allocated after every start of an application, a predicted value of the Java heap at the time of next start of the application is computed (102), and initial allocation of the Java heap for the application is performed on the basis of the predicted value of the Java heap when the application is started next time (103). The frequency of garbage collection during application start is effectively reduced without increasing total memory, thereby decreasing the time for application start and improving the smoothness of start animations.

(57) 摘要: 一种终端中应用启动时的内存分配方法及其系统，涉及终端应用技术。根据已记录的应用每次启动后实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值(102)，并根据该 Java 堆预测值在该应用下一次启动时为该应用初始分配 Java 堆(103)，在不增加内存总量的前提下，有效减少了应用启动期间的垃圾回收次数，从而减少应用启动时间，提高启动动画平滑度。

WO 2016/119605 A1

RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 包括国际检索报告(条约第 21 条(3))。

终端中应用启动时的内存分配方法及其系统

技术领域

本发明涉及终端应用技术，特别涉及终端中应用启动时的内存分配方法及其系统。

背景技术

在 android 或者类 android 系统中，应用启动时的用户体验主要体现在应用启动时间的长短和应用启动动画的平滑程度上。这两个方面都跟应用启动时的内存的大小和分配方法有关。在内存有限的情况下，对内存需求较高的应用比如游戏等可能会面临启动时间长、启动动画卡顿等用户体验问题。

在 android 系统里，每个应用在启动时，系统会为其分配一块大小为 S_0 的 Java 堆内存空间，用以存放程序在启动和运行时动态产生的 Java 类对象等相关数据。Java 堆是由垃圾回收器管理的，不再需要的对象数据会被垃圾回收器清除，清理出来的堆空间可以用来存放其它对象数据。

现有 android 系统无法预知每个应用所需 Java 堆空间的大小，为了不浪费内存，S_0 的取值会相对较小。如果应用在启动过程中产生大量对象数据，经过垃圾回收后仍不能满足分配需求，系统会动态增加该应用的堆空间容量。这样一来，在应用启动阶段，堆空间的大小会随着启动的推进而不断增加，总体呈现为递增序列 $\{S_0,S_1,S_2,S_3\ldots\}$。

本发明的发明人发现，对所有应用的每次启动，现有 android 系统都是从一个统一的较小值 S_0 开始，根据应用需求逐步增加 Java 堆空间大小。这样做的好处是不浪费内存，但是存在一个缺点，会降低应用启动时的用户体验。在应用启动阶段，这种机制会触发垃圾回收，垃圾回收是有时间代价的，它会造成启动时间的增加和启动动画的卡顿。

发明内容

本发明的目的在于提供一种终端中应用启动时的内存分配方法及其系统，在不增加内存总量的前提下，有效减少了应用启动期间的垃圾回收次数。

为解决上述技术问题，本发明的实施方式公开了一种终端中应用启动时的内存分配方法，该方法包括以下步骤：

记录应用每次启动后实际分配的 Java 堆大小；

根据已记录的应用的实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值；

根据 Java 堆预测值，在应用下一次启动时为该应用初始分配 Java 堆。

本发明的实施方式还公开了一种终端中应用启动时的内存分配系统，该系统包括：

记录模块，用于记录应用每次启动后实际分配的 Java 堆大小；

计算模块，用于根据记录模块已记录的应用的实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值；以及

分配模块，用于根据计算模块计算的 Java 堆预测值，在应用下一次启动时为该应用初始分配 Java 堆。

本发明实施方式与现有技术相比，主要区别及其效果在于：

在本发明中，根据已记录的应用每次启动后实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值，并根据该 Java 堆预测值在该应用下一次启动时为该应用初始分配 Java 堆，在不增加内存总量的前提下，有效减少了应用启动期间的垃圾回收次数，从而减少应用启动时间，提高启动动画平滑度。

进一步地，选取 c 为 0.4~0.6 之间的值，既可以保证每次启动的 Java 堆预测值不断逼近实际分配的 Java 堆大小，又可以避免每次启动时初始分配的 Java 堆大小的波动，以造成不必要的内存浪费。

进一步地，考虑更多次启动后实际分配的 Java 堆大小，使得下一次启动的 Java 堆预测值更准确。

附图说明

图 1 是本发明第一实施方式中一种终端中应用启动时的内存分配方法的流程示意图；

图 2 是 android 系统上典型的应用启动过程的示意图。

图 3 和图 4 分别是 c 取值为 0.8 和 0.2 时预测结果与 M 值的曲线图。

图 5 是本发明第二实施方式中一种终端中应用启动时的内存分配系统的结构示意图。

具体实施方式

在以下的叙述中，为了使读者更好地理解本申请而提出了许多技术细节。但是，本领域的普通技术人员可以理解，即使没有这些技术细节和基于以下各实施方式的种种变化和修改，也可以实现本申请各权利要求所要求保护的技术方案。

为使本发明的目的、技术方案和优点更加清楚，下面将结合附图对本发明的实施方式作进一步地详细描述。

本发明第一实施方式涉及一种终端中应用启动时的内存分配方法。图 1 是该终端中应用启动时的内存分配方法的流程示意图。可以理解，上述终端可以为智能手机、平板电脑、电视机、数码相机、游戏机等等。

如图 1 所示，该方法包括以下步骤:

在步骤 101 中，记录应用每次启动后实际分配的 Java 堆大小。

此后进入步骤 102，根据已记录的应用的实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值。

此后进入步骤 103，根据 Java 堆预测值，在应用下一次启动时为该应用初始分配 Java 堆。

此后结束本流程。

可选地，上述方法应用于安卓系统或类安卓系统。此外，可以理解，在本发明的其他实施方式中，上述方法也可以应用于使用 Java 语言的其他操作系统中。

在本实施方式的方法中，根据已记录的应用每次启动后实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值，并根据该 Java 堆预测值在该应用下一次启动时为该应用初始分配 Java 堆，在不增加内存总量的前提下，有效减少了应用启动期间的垃圾回收次数，从而减少应用启动时间，提高启动动画平滑度。

其中，在步骤 102 中可以以各种方式对该应用下一次启动的 Java 堆预测值进行计算，具体地说:

可以在步骤 102 中，根据 $S_{n-1} = c * S_{n-2} + c' * M_{n-2}$ 计算该应用第 n 次启动的 Java 堆预测值，n 为大于 1 的整数。

其中，S_{n-1} 为该应用第 n 次启动的 Java 堆预测值，S_{n-2} 为该应用第 n-1 次启动的 Java 堆预测值，M_{n-2} 为该应用第 n-1 次启动后实际分配的 Java 堆大小，$0<c<1$，$0<c'<1$，且 $c+c'=1$。

优选地，c 可以为 0.4~0.6 之间的值。选取 c 为 0.4~0.6 之间的值，既可以保证每次启动的 Java 堆预测值不断逼近实际分配的 Java 堆大小，又可以

避免每次启动时初始分配的 Java 堆大小的波动，以造成不必要的内存浪费。

需注意的是，当直接取 c=0 时，$S_{n-1} = M_{n-2}$，…，$S_2 = M_1$，$S_1 = M_0$，相当于直接按照上一次启动完成时 Java 堆的大小来为下一次启动分配堆。这样做，虽然可能会产生波动，但优化效果通常比较显著。

也可以在步骤 102 中把以往历次启动的 Java 堆大小都考虑进来，即根据 $S_{n-1} = c_0 * M_{n-2} + c_1 * M_{n-3} + \ldots + c_{n-2} * M_0$ 计算该应用第 n 次启动的 Java 堆预测值，n 为大于 1 的整数。

其中，S_{n-1} 为该应用第 n 次启动的 Java 堆预测值，M_0、M_1…，M_{n-2} 分别为该应用第 1 次、第 2 次…，第 n-1 次启动后实际分配的 Java 堆大小，$0< c_0, c_1\ldots, c_{n-2}<1$，且 $c_0+c_1+\ldots+c_{n-2}=1$。

可以理解，在本发明的各个实施方式中，可以对 M_0、M_1…，M_{n-2} 进行算术平均以得到 S_{n-1}，即 $c_0 = c_1 = \ldots = c_{n-2} = 1/(n-1)$，也可以对 M_0、M_1…，M_{n-2} 进行加权平均，即可以为 c_0，c_1…，c_{n-2} 选择不同的取值，在一优选实例中，选择 $c_0>c_1>\ldots>c_{n-2}$，以使得最近几次启动后实际分配的 Java 堆大小的权重更大，或使用其他函数来进行计算。

此外，可以理解，在本发明的其他实施方式中，也可以根据需要仅考虑前两次、前三次或前 m 次（m 为整数）启动后实际分配的 Java 堆大小以对下一次启动的 Java 堆预测值进行计算。

考虑更多次启动后实际分配的 Java 堆大小，可以使下一次启动的 Java 堆预测值更准确。

在实际操作时，可以根据需要选择上述计算方式中的一种或其组合。例如在启动该应用的前几次采用前一种方式，之后采用后一种方式，以达到更佳的效果。

此外，可以理解，在本发明的其他实施方式中，也可以不根据历史数据

来计算上述应用下一次启动的 Java 堆预测值，在安装应用的同时即通过分析应用本身的内存使用情况来计算该应用启动的 Java 堆预测值。

由上可以看到，为了减少垃圾回收次数，提升应用启动时的用户体验，本发明提出了一种新的内存分配方法：预测每一个应用在下次启动时所需最大 Java 堆大小 M，当用户下次启动该应用时，系统将直接分配大小为 M 的堆给该应用。这样，应用启动期间发生垃圾回收的次数就会减少，从而减少了应用启动时间和启动动画的卡顿。

我们在此对该预测方法作几点说明。首先，上述预测方法针对的是每一个应用。因为每个应用在启动时所需最大 Java 堆大小都不相同，本发明针对每个应用预测出的最大堆大小也是不同的。其次，上述预测方法预测的是应用启动期间所需堆大小。最后，每个应用的启动时间长短不一，所以需要界定“启动期间”。如图 2 所示，以 android 系统上典型的应用启动过程为例，本发明所使用的“启动期间”包括从“1 用户点击应用启动图标”到“4 应用第一帧显示”共四个步骤。

以下将进一步以安卓系统和 $S_{n-1} = c * S_{n-2} + c' * M_{n-2}$ 为例对上述方法进行详细描述。具体如下：

（1）应用第一次启动时采用与现有 android 系统相同的堆分配方法，即先分配大小为 S_0 的 Java 堆，其中 S_0 是一个统一的较小值，应用在启动过程中会动态增加堆空间容量。当启动过程完成时，系统会收到“Displayed”消息，表示应用第一帧已经显示。此时我们记下应用当前 Java 堆大小 M_0。

（2）应用第二次启动时，直接分配大小为 S_1 的 Java 堆，$S_1 = c * S_0 + c' * M_0$（其中 $0<c<1$，$0<c'<1$，且 $c+c'=1$），通常情况下 c 取 0~1 中间值 0.5，也可以根据实际需要选择 0~1 之间的其他取值。当启动过程完成时，记下当前 Java 堆大小 M_1。

（3）依次类推，应用第 n 次启动时，直接分配大小为 S_{n-1} 的 Java 堆，

$S_{n-1} = c * S_{n-2} + c' * M_{n-2}$（其中 0<c<1，0<c'<1，且 c+c'=1）。当启动过程完成时，记下当前 Java 堆大小 M_{n-1}。

下面将对 c 的取值稍作说明，以了解 c 的取值对预测的影响。如果 c=1，则 $S_{n-1} = S_{n-2} = \ldots = S_1 = S_0$，上述预测方法将退化成现有 android 系统所使用的堆分配方法；如果 c=0，则 $S_{n-1} = M_{n-2}$，…，$S_2 = M_1$，$S_1 = M_0$，上述预测方法相当于直接按照上一次启动完成时 Java 堆的大小来为下一次启动分配堆。因为应用每次启动完成时 Java 堆的大小可能并不相同，这跟 S_0 的取值有关，也跟应用本身的实现有关（例如应用更新），如果直接取 c=0，可能会造成 S_n 随着 M_{n-1} 波动。优选地，在本实施例中，c 取 0~1 中间值 0.5，这样既可以保证预测结果不断逼近真实值，也可以避免预测结果不必要的波动。

此外，也可以根据系统的设计倾向对 c 取 0~1 之间其它值。如图 3 和图 4 所示，当 c 越接近 1 时（例如 c=0.8），预测结果逼近 M 值越慢；当 c 越接近 0 时（例如 c=0.2），预测结果逼近 M 值越快，但是波动更大，即如图 4 所示，在第 3 次和第 4 次启动时预测结果大于 M 值，在第 5 次和第 6 次启动时预测结果小于 M 值，则在预测结果大于 M 值时会造成一定的内存浪费。因此，我们可以根据系统的设计倾向对 c 进行灵活取值。如果追求更快见效，可以使用接近 0 的取值；如果追求稳定性，可以使用接近 1 的取值。

因此，本发明提出了一种新的内存分配策略，可以在不增加内存总量的前提下，更加合理的分配内存，从而减少应用启动时间，提高启动动画平滑度。

可以理解，以上仅为一优选例，在本发明的其他实施例中，也可以根据需要使用其他计算方式或其组合，并应用于使用 Java 语言的其他操作系统中。

本发明的各方法实施方式均可以以软件、硬件、固件等方式实现。不管本发明是以软件、硬件、还是固件方式实现，指令代码都可以存储在任何类

型的计算机可访问的存储器中（例如永久的或者可修改的，易失性的或者非易失性的，固态的或者非固态的，固定的或者可更换的介质等等）。同样，存储器可以例如是可编程阵列逻辑（Programmable Array Logic，简称“PAL”）、随机存取存储器（Random Access Memory，简称“RAM”）、可编程只读存储器（Programmable Read Only Memory，简称“PROM”）、只读存储器（Read-Only Memory，简称“ROM”）、电可擦除可编程只读存储器（Electrically Erasable Programmable ROM，简称“EEPROM”）、磁盘、光盘、数字通用光盘（Digital Versatile Disc，简称“DVD”）等等。

本发明第二实施方式涉及一种终端中应用启动时的内存分配系统。图 5 是该终端中应用启动时的内存分配系统的结构示意图。如图 5 所示，该系统包括:

记录模块，用于记录应用每次启动后实际分配的 Java 堆大小。

计算模块，用于根据记录模块已记录的应用的实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值。以及

分配模块，用于根据计算模块计算的 Java 堆预测值，在应用下一次启动时为该应用初始分配 Java 堆。

可选地，上述系统应用于安卓系统或类安卓系统。此外，可以理解，在本发明的其他实施方式中，上述系统也可以应用于使用 Java 语言的其他操作系统中。

在本实施方式的系统中，计算模块根据已记录的应用每次启动后实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值，分配模块根据该 Java 堆预测值在该应用下一次启动时为该应用初始分配 Java 堆，在不增加内存总量的前提下，有效减少了应用启动期间的垃圾回收次数，从而减少应用启动时间，提高启动动画平滑度。

其中，上述计算模块可以以各种方式对该应用下一次启动的 Java 堆预测

值进行计算，具体地说：

上述计算模块可以用于根据 $S_{n-1} = c * S_{n-2} + c' * M_{n-2}$ 来计算应用第 n 次启动的 Java 堆预测值，n 为大于 1 的整数。

其中，S_{n-1} 为应用第 n 次启动的 Java 堆预测值，S_{n-2} 为应用第 n-1 次启动的 Java 堆预测值，M_{n-2} 为应用第 n-1 次启动后实际分配的 Java 堆大小，$0<c<1$，$0<c'<1$，且 $c+c'=1$。

优选地，c 为 0.4~0.6。选取 c 为 0.4~0.6，既可以保证每次启动的 Java 堆预测值不断逼近实际分配的 Java 堆大小，又可以避免每次启动时初始分配的 Java 堆大小的波动，以造成不必要的内存浪费。

需注意的是，当直接取 c=0 时，$S_{n-1} = M_{n-2}$，…，$S_2 = M_1$，$S_1 = M_0$，相当于直接按照上一次启动完成时 Java 堆的大小来为下一次启动分配堆。这样做，虽然可能会产生波动，但优化效果通常比较显著。

上述计算模块也可以用于根据 $S_{n-1} = c_0 * M_{n-2} + c_1 * M_{n-3} + \ldots + c_{n-2} * M_0$ 来计算应用第 n 次启动的 Java 堆预测值。

其中，S_{n-1} 为应用第 n 次启动的 Java 堆预测值，M_0、M_1…，M_{n-2} 分别为应用第 1 次、第 2 次…，第 n-1 次启动后实际分配的 Java 堆大小，$0< c_0$，c_1…，$c_{n-2}<1$，且 $c_0+c_1+\ldots+c_{n-2}=1$。

可以理解，在本发明的各个实施方式中，可以对 M_0、M_1…，M_{n-2} 进行算术平均以得到 S_{n-1}，即 $c_0 = c_1 = \ldots = c_{n-2} = 1/(n-1)$，也可以对 M_0、M_1…，M_{n-2} 进行加权平均，即可以为 c_0，c_1…，c_{n-2} 选择不同的取值，在一优选实例中，选择 $c_0>c_1>\ldots>c_{n-2}$，以使得最近几次启动后实际分配的 Java 堆大小的权重更大，或使用其他函数来进行计算。

此外，可以理解，在本发明的其他实施方式中，也可以根据需要仅考虑前两次、前三次或前 m 次（m 为整数）启动后实际分配的 Java 堆大小对下

一次启动的 Java 堆预测值进行计算。

考虑更多次启动后实际分配的 Java 堆大小，可以使下一次启动的 Java 堆预测值更准确。

在实际操作时，可以根据需要选择上述计算方式中的一种或其组合。例如在启动该应用的前几次采用前一种方式，之后采用后一种方式，以达到更佳的效果。

此外，可以理解，在本发明的其他实施方式中，也可以不根据历史数据来计算上述应用下一次启动的 Java 堆预测值，在安装应用的同时即通过分析应用本身的内存使用情况来计算该应用启动的 Java 堆预测值。

第一实施方式是与本实施方式相对应的方法实施方式，本实施方式可与第一实施方式互相配合实施。第一实施方式中提到的相关技术细节在本实施方式中依然有效，为了减少重复，这里不再赘述。相应地，本实施方式中提到的相关技术细节也可应用在第一实施方式中。

需要说明的是，本发明各设备实施方式中提到的各模块都是逻辑模块，在物理上，一个逻辑模块可以是一个物理模块，也可以是一个物理模块的一部分，还可以以多个物理模块的组合实现，这些逻辑模块本身的物理实现方式并不是最重要的，这些逻辑模块所实现的功能的组合才是解决本发明所提出的技术问题的关键。此外，为了突出本发明的创新部分，本发明上述各设备实施方式并没有将与解决本发明所提出的技术问题关系不太密切的模块引入，这并不表明上述设备实施方式并不存在其它的模块。

需要说明的是，在本专利的权利要求和说明书中，诸如第一和第二等之类的关系术语仅仅用来将一个实体或者操作与另一个实体或操作区分开来，而不一定要求或者暗示这些实体或操作之间存在任何这种实际的关系或者顺序。而且，术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含，从而使得包括一系列要素的过程、方法、物品或者设备不仅包括那些要

素，而且还包括没有明确列出的其他要素，或者是还包括为这种过程、方法、物品或者设备所固有的要素。在没有更多限制的情况下，由语句“包括一个”限定的要素，并不排除在包括所述要素的过程、方法、物品或者设备中还存在另外的相同要素。

虽然通过参照本发明的某些优选实施方式，已经对本发明进行了图示和描述，但本领域的普通技术人员应该明白，可以在形式上和细节上对其作各种改变，而不偏离本发明的精神和范围。

权 利 要 求 书

1. 一种终端中应用启动时的内存分配方法，其特征在于，该方法包括以下步骤:

记录应用每次启动后实际分配的 Java 堆大小;

根据已记录的所述应用的实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值;

根据所述 Java 堆预测值，在所述应用下一次启动时为该应用初始分配 Java 堆。

2. 根据权利要求 1 所述的终端中应用启动时的内存分配方法，其特征在于，所述“根据已记录的所述应用的实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值”的步骤中，根据 $S_{n-1} = c * S_{n-2} + c' * M_{n-2}$ 计算所述应用第 n 次启动的 Java 堆预测值，n 为大于 1 的整数;

其中，S_{n-1} 为所述应用第 n 次启动的 Java 堆预测值，S_{n-2} 为所述应用第 n-1 次启动的 Java 堆预测值，M_{n-2} 为所述应用第 n-1 次启动后实际分配的 Java 堆大小，$0<c<1$，$0<c'<1$，且 $c+c'=1$。

3. 根据权利要求 2 所述的终端中应用启动时的内存分配方法，其特征在于，c 为 0.4~0.6 之间的任一值。

4. 根据权利要求 1 所述的终端中应用启动时的内存分配方法，其特征在于，所述“根据已记录的所述应用的实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值”的步骤中，根据 $S_{n-1} = c_0 * M_{n-2} + c_1 * M_{n-3} + \ldots + c_{n-2} * M_0$ 计算所述应用第 n 次启动的 Java 堆预测值，n 为大于 1 的整数;

其中，S_{n-1} 为所述应用第 n 次启动的 Java 堆预测值，M_0、M_1...，M_{n-2} 分别为所述应用第 1 次、第 2 次...，第 n-1 次启动后实际分配的 Java 堆大小，

$0< c_0, c_1..., c_{n-2}<1$，且 $c_0+c_1+...+c_{n-2}=1$。

5. 根据权利要求 1 至 4 中任一项所述的终端中应用启动时的内存分配方法，其特征在于，上述方法应用于安卓系统。

6. 一种终端中应用启动时的内存分配系统，其特征在于，该系统包括：

记录模块，用于记录应用每次启动后实际分配的 Java 堆大小；

计算模块，用于根据所述记录模块已记录的所述应用的实际分配的 Java 堆大小计算该应用下一次启动的 Java 堆预测值；以及

分配模块，用于根据所述计算模块计算的所述 Java 堆预测值，在所述应用下一次启动时为该应用初始分配 Java 堆。

7. 根据权利要求 6 所述的终端中应用启动时的内存分配系统，其特征在于，所述计算模块用于根据 $S_{n-1} = c * S_{n-2} + c' * M_{n-2}$ 来计算所述应用第 n 次启动的 Java 堆预测值，n 为大于 1 的整数；

其中，S_{n-1} 为所述应用第 n 次启动的 Java 堆预测值，S_{n-2} 为所述应用第 n-1 次启动的 Java 堆预测值，M_{n-2} 为所述应用第 n-1 次启动后实际分配的 Java 堆大小，$0<c<1$，$0<c'<1$，且 $c+c'=1$。

8. 根据权利要求 7 所述的终端中应用启动时的内存分配系统，其特征在于，c 为 0.4~0.6 之间的任一值。

9. 根据权利要求 6 所述的终端中应用启动时的内存分配系统，其特征在于，所述计算模块用于根据 $S_{n-1} = c_0 * M_{n-2} + c_1 * M_{n-3} + ...+ c_{n-2} * M_0$ 来计算所述应用第 n 次启动的 Java 堆预测值，n 为大于 1 的整数；

其中，S_{n-1} 为所述应用第 n 次启动的 Java 堆预测值，M_0、$M_1...$，M_{n-2} 分别为所述应用第 1 次、第 2 次...，第 n-1 次启动后实际分配的 Java 堆大小，$0< c_0, c_1..., c_{n-2}<1$，且 $c_0+c_1+...+c_{n-2}=1$。

10. 根据权利要求 6 至 9 中任一项所述的终端中应用启动时的内存分配

系统，其特征在于，上述系统应用于安卓系统。

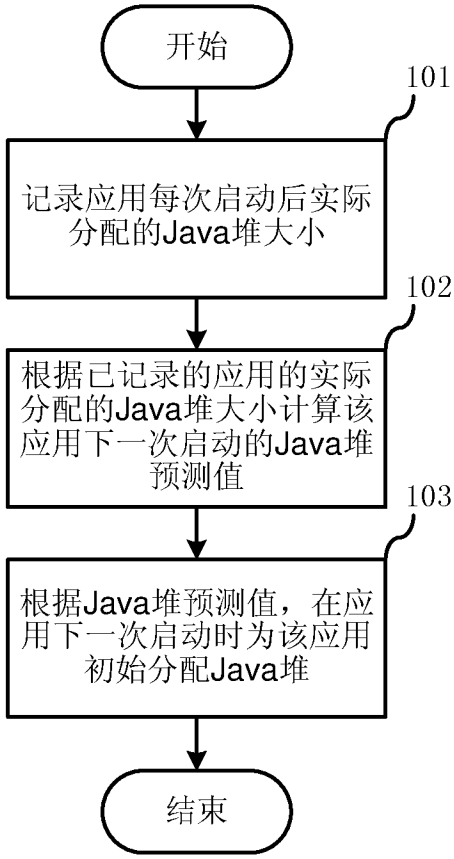

开始
101
记录应用每次启动后实际分配的Java堆大小
102
根据已记录的应用的实际分配的Java堆大小计算该应用下一次启动的Java堆预测值
103
根据Java堆预测值，在应用下一次启动时为该应用初始分配Java堆
结束

图 1

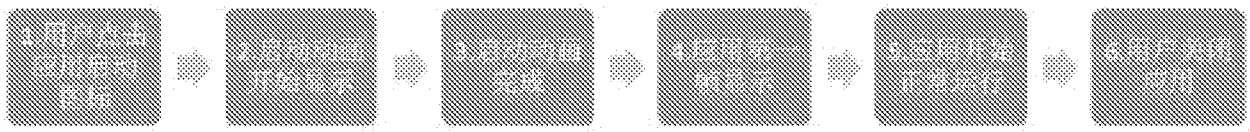


图 2

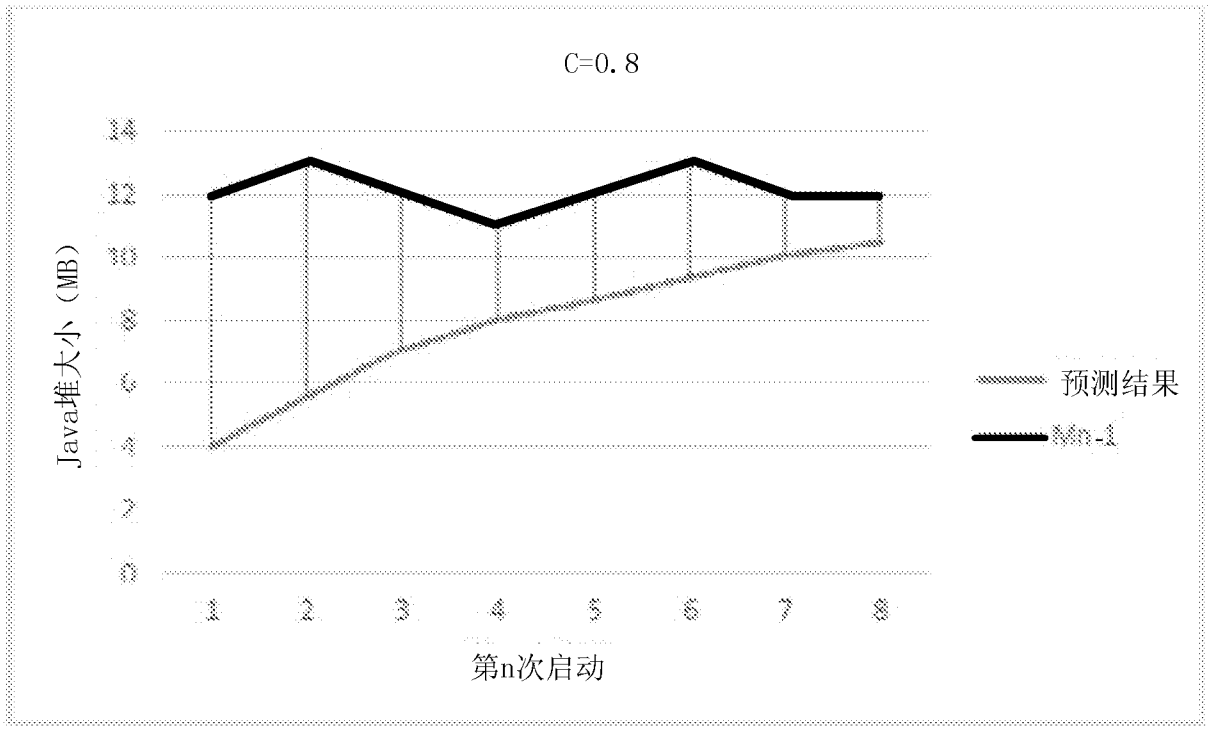

C=0.8
Java堆大小（MB）
0
2
4
6
8
10
12
14
1
2
3
4
5
6
7
8
第n次启动
预测结果
Mn-1

图 3

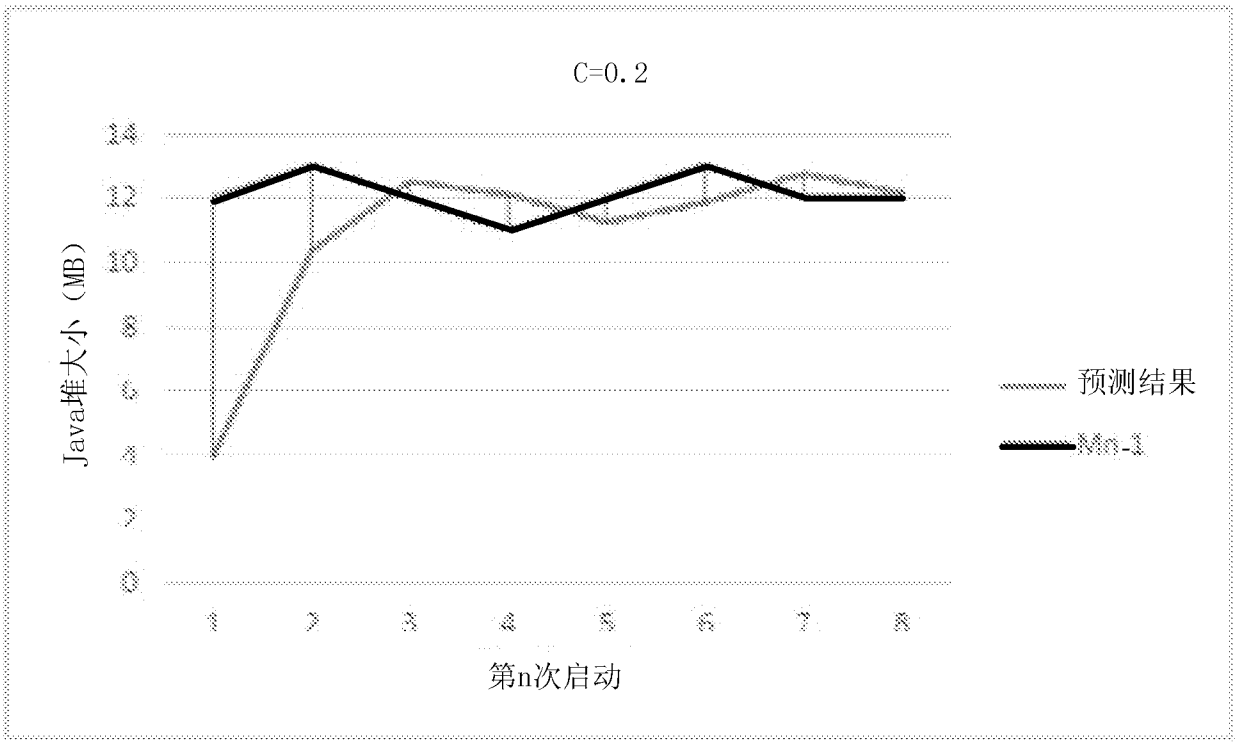

C=0.2
Java堆大小（MB）
0
2
4
6
8
10
12
14
1
2
3
4
5
6
7
8
第n次启动
预测结果
Mn-1

图 4

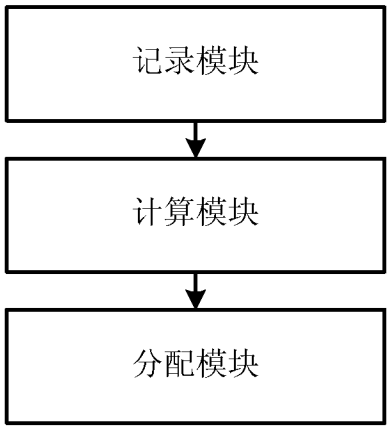


图 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2016/071165

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/445 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNPAT, CNKI, WPI, EPODOC: program, software, application, java heap, memory, allocate, speed, time, start, apply, start, speed, time, history, forecast, estimate, initial

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 103927264 A (CHINESE AERONAUTICAL RADIO ELECTRONICS RESEARCH INSTITUTE) 16 July 2014 (16.07.2014) description, paragraphs [0029]-[0036]	1-10
A	JP 2012032896 A (FUJITSU FRONTECH LTD.) 16 February 2012 (16.02.2012) the whole dcoument	1-10
A	CN 103544055 A (SHENZHEN TCL NEW TECHNOLOGY CO., LTD.) 29 January 2014 (29.01.2014) the whole dcoument	1-10
A	CN 103617060 A (INTEGRATED ELECTRONIC SYSTEMS LAB CO., LTD.) 05 March 2014 (05.03.2014) the whole dcoument	1-10

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&"document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
17 March 2016	12 April 2016
Name and mailing address of the ISA State Intellectual Property Office of the P. R. China No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088, China Facsimile No. (86-10) 62019451	Authorized officer SUN, Guohui Telephone No. (86-10) 62413599

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2016/071165

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
CN 103927264 A	16 July 2014	None	
JP 2012032896 A	16 February 2012	None	
CN 103544055 A	29 January 2014	None	
CN 103617060 A	05 March 2014	None	

国际检索报告

国际申请号
PCT/CN2016/071165

A. 主题的分类

G06F 9/445(2006.01)i

按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

G06F

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称，和使用的检索词（如使用））

CNPAT,CNKI,WPI,EPODOC:程序，应用，软件，启动，速度，时间，java堆，内存，分配，历史，预测，预计，预估，估计，初始，program, software, application, java heap, memory, allocate, speed, time, start

C. 相关文件

类 型*	引用文件，必要时，指明相关段落	相关的权利要求
A	CN 103927264 A (中国航空无线电电子研究所) 2014年 7月 16日 (2014-07-16) 说明书第[0029]-[0036]段	1-10
A	JP 2012032896 A (FUJITSU FRONTECH LTD.) 2012年 2月 16日 (2012-02-16) 全文	1-10
A	CN 103544055 A (深圳TCL新技术有限公司) 2014年 1月 29日 (2014-01-29) 全文	1-10
A	CN 103617060 A (积成电子股份有限公司) 2014年 3月 5日 (2014-03-05) 全文	1-10

☐ 其余文件在C栏的续页中列出。

☑ 见同族专利附件。

* 引用文件的具体类型：

“A” 认为不特别相关的表示了现有技术一般状态的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件，或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件（如具体说明的）

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布，与申请不相抵触，但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件，单独考虑该文件，认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件，当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时，要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期	国际检索报告邮寄日期
2016年 3月 17日	2016年 4月 12日

ISA/CN的名称和邮寄地址	受权官员
中华人民共和国国家知识产权局(ISA/CN) 中国北京市海淀区蓟门桥西土城路6号 100088	孙国辉
传真号 (86-10)62019451	电话号码 (86-10)62413599

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2016/071165

检索报告引用的专利文件			公布日 (年/月/日)	同族专利	公布日 (年/月/日)
CN	103927264	A	2014年 7月 16日	无	
JP	2012032896	A	2012年 2月 16日	无	
CN	103544055	A	2014年 1月 29日	无	
CN	103617060	A	2014年 3月 5日	无	