文章编号:1671-4598(2017)08-0162-04 DOI:10.16526/j.cnki.11—4762/tp.2017.08.042 中图分类号:TP391 文献标识码:A

基于 Open VG 云电子书系统的多级优化框架设计

张春燕,于 丽
(新疆警察学院 信息安全工程系,乌鲁木齐 830000)

摘要:针对电子书应用存在的文件格式、性能效率低下和图像失真等问题,设计了一种应用于云电子书系统的多级优化框架,优化框架主要体现在如下三个方面:第一,对向量图形类库的性能进行描述,并提出了一种优化算法,减少了类库的时间复杂度;第二,在嵌入式 GPU 上并行进行坐标系统的计算;利用 GPU 在并行计算方面的优势,云电子书在向量图形类库方面获取了显著的性能提升;第三,云电子书将文件转化功能转嫁给 Hadoop 云平台,节省了移动设备的能量消耗和计算时间。同时为了对 Hadoop 调度过程中的数据位置问题进行优化,将位置感知调度器运用到提出的系统;实验结果表明:云电子书系统与最初的 Open VG 类库相比,性能提升了约 70%,而且云电子书系统与连续服务器平台相比,计算时间减小了约 60%。

关键词:电子书;向量图形类库;GPU;云;位置感知调度器

Multiple—level Optimization Framework Design of Cloud Electronic Book System Based on Open VG

Zhang Chunyan, Yu Li
(Department of Information Security Engineering, Xinjiang Police College, Urumqi 830000, China)

Abstract: Concerning the problem of file format, low performance efficiency and image distortion in the application of e—books, a multiple—level optimization framework of cloud e—book is proposed, in which the optimization framework is mainly reflected in the following three aspects. Firstly, the performance of the vector graphics library is described, and an optimization algorithm is proposed to reduce the time complexity of the class library. Secondly, parallel coordinate system is used to calculate on an embedded GPU. Cloud electronic books in the vector graphics library has obtained a significant performance improvement by making use of the advantage of GPU in parallel computing. Thirdly, file conversion function of cloud e—books is passed on to the Hadoop cloud platform, by which cloud save energy consumption and computation time of the mobile equipment. And in order to optimize the Hadoop scheduler in the process of data location problem, location aware scheduler is applied to the proposed system. The experimental results show that, compared with the original VG Open class library, the performance of the cloud e—book system is improved by about 70%, and the computing time is reduced by about 60% compared with the continuous server platform.

Keywords: electronic books; vector graphic library; GPU; cloud; location aware scheduler

0 引言

电子书[1]主要由数字形式的文本和图像构成,可在计算机或其他电子设备上观看。最近十多年,许多电子书应用程序可以在移动设备上运行,而且出现了很多电子书应用程序。但是,这些应用程序没有得到广泛应用,这是因为电子书应用在移动设备上运行存在一些缺陷。其最大问题是图像失真,当用户调节图像尺寸时,电子书的内容和图像可能发生失真[2],这是因为电子书上显示的大多数图像都是位图类型的图形。针对该问题,可缩放向量图像(Scalable Vector Graphics, SVG)技术[3]是解决图像失真的有效途径。

近些年,一些工业制造商已经对电子书应用进行了广泛的研究。大部分研究工作关注的是电子书应用[4]、安全[5-6]、和平台标准化[7]。也有一些研究者[8-9]在移动操作系统上采用 SVG 技术,这些移动操作系统有 Android、微软的 Windows Phone(现在已经很少见到 Windows 操作系统的智能手机)和苹果的 iOS。文献[10-11]通过修改光栅化的处理过程,利用双扫描线对渲染过程进行加速,进而优化了 Open VG[3]类库的性能,其主要关注于渲染和光栅化过程中的算法优化问题。但是,在 Open VG[3]类库中路径转化是另外一个计算强度函数,而 Android VG[12]是一种三角划分过程。

一般情况下,向量图形技术不仅可以解决图像失真问题,而且还可以很好地解决电子书中文件格式统一的问题。但是,将向量图形技术引入到电子书应用中后,产生了另外两个问题:

1)向量图形图像在电子书应用中所表现出的性能低于一张由位图进行渲染的图像,这是因为向量图形技术需要花费大量的时间处理路径的技术和绘制。

2)向量图形图像的文件尺寸与以位图格式存储的图像相比要大很多,这是因为向量信息需要复杂的描述。

为了解决上述问题,本文提出了一种优化电子书应用,即将嵌入式 GPU[13]和云平台的计算资源进行了有效的整合。将 Open VG[3]类库和 Android 系统进行了整合,并且通过嵌入式 GPU 加速优化了 Android 系统上的向量图形类库。其突出的优点有:

1)通过向量图形库解决了电子书应用中存在的图像失真问题;

2)云电子书可以将向量图形库的时间复杂度从 $O(n^2)$ 减

收稿日期:2017-02-14; **修回日期**:2017-03-06。

作者简介:张春燕(1979-),女,江苏丰县人,硕士,讲师,主要从事算法设计、控制理论与控制工程等方向的研究。

少到 $O(n*\log n)$，因此通过修改图形三角划分算法以及将路径计算任务转嫁到嵌入式 GPU；

3）云电子书在 Hadoop 平台[14]上开发并执行了一个位置感知调度器，该调度器不仅提高了向量图形转化器的性能，而且提高了 Hadoop 平台上的数据命中率。

1　云电子书的优化

本节介绍云电子书系统中的系统类库优化过程，即基于 Open VG[3]规范和源代码开发出了优化类库。

为了识别和克服瓶颈，对计算密集型阶段进行了性能分析，这里的计算密集型阶段指的是：Open VG 流程中的扁平化、转化、三角划分和光栅化。为了对性能进行分析，将性能分析函数置入 Open VG 类库。三个广为人知的简单应用包括：Tiger VG、Apple VG 和 Star VG，这三个应用在改进的向量图形类库中执行。其中，三角划分是 Open VG 类库中的性能瓶颈。三角划分过程含有许多图形计算任务，如计算顶点的坐标，并将整个图形分割成许多三角形。因此，优化三角划分处理过程是向量图形类库中最重要的问题。

1.1　提出的加速三角划分算法

加速三角划分算法主要聚焦于对计算步骤的改进以及减少三角划分算法的时间复杂度。通过将三角划分步骤简化为绘画步骤，进而简化了计算步骤。

Shiva VG 的计算过程中存在三个步骤。第一个步骤是计算模板缓存中的绘制图形，然后在帧缓存中寻找绘制区域以添加颜料。Android VG 在三角划分和帧缓存间存在一个更加复杂的流程，Android VG 对平面图形进行了三角划分，并且向深度缓存中输出了各部分的划分图形。深度缓存步骤结束之后，Android VG 就具有了与 Shiva VG 相同的数据流。与上述向量图形类库不同的是，云电子书已经对计算步骤进行了简化。于是云电子书在三角划分步骤中遵循相同的步骤，但是，云电子书直接输出了与绘制设置相关的划分信息给帧缓存。这样，云电子书的计算步骤降低了时间消耗，这是因为对三角划分步骤进行了改进，这里的时间消耗是指不同缓存间数据移动所产生的消耗。

向量图形类库中原始的三角划分算法的时间复杂度为 $O(n^2)$，其中 n 表示顶点的个数。如果输入的资源是一幅复杂的照片，比如 Tiger VG，三角划分步骤所消耗的时间是向量图形类库的瓶颈。为了减小三角划分步骤的时间复杂度，本文中利用一种单调三角划分算法替代原始三角划分算法。将一个多边形拼接成多个单调多边形的算法如算法 1 中所述。

定义 1：x 单调多边形。如果每一行正交轴 x 轴与 P 最多相交两次，那么这个多边形 P 称为 x 单调。

算法 1：预处理单调图形算法

输入：一个多边形 $G=(V,E)$，$V=\{v_1,v_2,\ldots,v_n\}$ 和一个凹顶点 $C_V=\varphi$ 的空集合

输出：单调图形 M 的一个集合

1　for $j=1$；$j\leqslant n$ do
2　if v_j 是一个凹顶点 then
3　将 v_j 加入到 C_V 中；
4　end
5　end
6　根据 C_V 的 x 坐标将 C_V 从左到右进行排序；
7　While 不是 C_V 的终点 do
8　p = C_V 中取出的第一个元素；
9　画一条垂直 x 轴且穿过凹顶点的直线 L；
10　找到一个与直线 L 和图像 G 的感兴趣边相邻的顶点集合 S；
11　if p 是一个拼接点 then
12　将 p 与最近的顶点相连接，这个顶点位于 p 的右侧，而且位于 S 中；
13　end
14　else
15　将 p 与最近的顶点相连接，这个顶点位于 p 的左侧，而且位于 S 中；
16　end
17　将含有顶点 p 且不含有任何其它凹顶点的子图形添加到 M 中；
18　End
19　返回 M；

预处理阶段的时间复杂度为 $O(n*\log n)$，因此，预处理阶段不能增加整个三角划分过程的时间复杂度。预处理阶段之后，输入的多边形已经转化成多个单调多边形块。在多个单调多边形上执行这种单调三角划分算法可以将三角划分阶段的时间复杂度从 $O(n^2)$ 降低为 $O(m*n\log n)$，其中 m 表示 G 中单调多边形的个数，m 的大小远小于 n。单调三角划分算法的具体过程如算法 2 中所示。

算法 2：单调三角划分算法

输入：一个单调图形 $G=(V,E)$ 和 $V=\{v_1,v_2,\ldots,v_n\}$

输出：一个三角图形 T

1　对 V 按照其 x 轴坐标将其从左至右进行排序
2　for $j=1;j\leqslant n$ do
3　if v_j 与 v_{j-1} 并不相邻 then
4　在 v_j 和堆上所有的顶点之间添加斜线；
5　从堆中取出所有的顶点；
6　将 v_{j-1} 和 v_j 压入到堆中；
7　end
8　else if v_j 具有与 v_{j-1} 一样的链路 then
9　for $k=1;k\leqslant$ 堆尺寸 do
10　从堆中获取 u；
11　if $v_iu_ku_{k-1}\leqslant\pi$ then
12　在 v_i 与 u_{i-1} 之间添加斜边；
13　从堆中取出 u_i；
14　end
15　向堆中压入 v_j
16　end
17　end
18　end

1.2　并行计算框架的优化

为了充分利用移动设备上的计算资源，本文提出了一种采用异构计算机机制的高性能框架，由于目前大多数移动设备都

配备一个嵌入式 GPU 单元，因此在 GPU 单元的协助下，这些应用的性能有所提高。一个 GPU 含有许多 ALU 单元，这些 ALU 单元能够并行执行程序，因此 GPU 在运行计算密集型应用时速度快于 CPU。根据 GPU 的特性，通过渲染脚本编程接口将移动设备平台上的三角划分计算任务转嫁到嵌入式 GPU 内进行处理。图 1 中给出了云电子书系统中转嫁机制的框架。

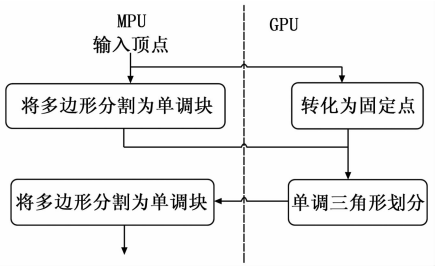

图 1 向量图形类库的并行框架

为了充分利用移动设备中的计算单元，转嫁机制将三角划分阶段的任务划分成 4 个子任务，并且在微型处理器单元（micro processing unit，MPU）和 GPU 上单独执行各个子任务。GPU 执行的任务属于计算密集型且高并行处理的任务，MPU 不能并行处理任务。这种机制不但利用了 GPU 的优势，而且也没有浪费 MPU 的计算资源。

2 云电子书的部署

云平台在云电子书系统中所扮演的角色是提供扩展资源，云平台由三个组件构成：Hadoop 分布式文件系统[14]（Hadoop Distributed File System，HDFS）、向量图形转化器程序和位置感知工作调度算法。由于云的分布式环境，本文提出了一种位置感知工作调度算法，并且将这种算法应用到云平台中，进而减少计算节点和数据节点间进行数据传输的时间复杂度。利用基于云电子书系统的优化云端可以解决移动设备中有限资源所引起的问题。

2.1 参数描述

位置感知调度器中的参数包括数据节点、数据的权重、输入数据的副本个数和每个节点上映射时隙点的个数。表 1 中给出了位置感知调度器中所有参数的简单定义。

表 1 位置感知调度器中定义的参数

符号	描述
n_j	n_j 表示一个节点号为 j 的数据节点。
N	N 表示云中的一个数据节点集合，$N=\{n_1,n_2,\ldots,n_p\}$。
$F(n_j)$	$F(n_j)$ 表示第 j 个节点上映射自由时隙个数。
d_i	d_i 表示数据编号为 i 的数据，每个 d_i 都是独特的。
$S(n_j,d_i)$	$S(n_j,d_i)$ 表示节点 n_j 中数据 d_i 的个数。
D	D 表示 MapReduce 中输入数据的集合，$D=\{d_1,d_2,\ldots,d_m\}$，在 HDFS 中将每个数据进行复制送给不同的节点。
$w(d_i)$	w 表示 MapReduce 中数据 d_i 的权重。
$W(n_j,T_{d_o})$	$W(n_j,T_{d_o})$ 表示当一个工作在时刻 T 请求数据时，节点 n_j 上数据干扰的权重。
T_{d_o}	T_{d_o} 表示一个携带请求数据 d_o 的工作到达的时间。

位置感知调度器主要关注“稀有资源”的配置。将数据节点称为稀有资源，数据节点中含有一些数据，这些资源相对缺乏，这是因为数据节点的大部分都属于所需数据，一些具有高优先级的任务占据这些数据节点。为了对稀有资源进行区分，将参数 $W(n_j,T_{d_o})$ 作为主要因子以测量位置感知调度器中数据节点的稀有程度。一个数据节点具有最小值的 $W(n_j,T_{d_o})$，即这个数据节点不属于稀有资源，因此调度器可以向这个数据节点派发计算任务。权重公式的表达式如式（1）所示：

$$W(n_j,T_{d_o})=\begin{cases}\sum_{i=1}^{m,i\neq 0}\left(\dfrac{S(n_j,d_i)*F(n_j)}{w(d_i)}\right) & \text{如果 } n_j \text{ 不含有 } d_i, \text{ 或者 } n_j \text{ 中不含有自由槽点}\\ \infty & \text{其它}\end{cases} \tag{1}$$

其中：

$$w(d_i)=\sum_{j=0}^{p}(S(n_j,d_i)*F(n_j)) \tag{2}$$

式（1）中，参数 $W(n_j,T_{d_o})$ 表示节点 n_j 中可利用数据 d_i 与云平台中可利用数据 $w(d_i)$ 总和之间的比值。因此，当节点 n_j 中含有一些数据时，参数 $W(n_j,T_{d_o})$ 的值会增加，且这些数据仅存在于节点 n_j 中。

2.2 位置感知调度算法

本文在工作调度器中采用位置感知调度器。在分配任务之前，工作跟踪器利用式（1）计算出每个节点上数据干扰的权重，每个节点利用空时隙点执行该任务。然后，工作跟踪器挑选出具有最小数据干扰权重的节点，并将任务分派给这个数据节点的任务跟踪器序列。算法 3 中给出了位置感知算法详细的步骤。调度器不仅可以计算数据干扰的权重，进而以一种简单的方式避免稀有资源分配，而且通过引入数据干扰权重概念增强了 Map Reduce 框架中的数据位置。因此，位置感知调度器减少了数据传输过程中的网络延迟引起的开销。另外，利用位置感知调度器还可以提高云平台的性能。

算法 3：位置感知调度算法

输入：一个用户任务 t 和 N 中的元数据

输出：将 t 分派给 n_t

1 计算 N 中每个数据节点的权重值 $w(d_i)$；

2 按照升序对数据节点的权重进行排序；

3 从 N 挑选出第一个具有最小数据干扰权重值的节点 n_{first}；

4 将任务 j 分派给节点 n_{first} 中的任务序列。

3 实验结果与分析

本节对 Android 系统和云平台上执行云电子书应用的实验结果进行了讨论。为了展现出云电子书性能的改善水平，本文在云电子书上运行了一系列的位图格式文件，并且与其它方法进行了对比实验。向量图形类库方法指的是 Open VG[3]、Android VG[12]，Hadoop 调度器的方法中包括先到先服务准则、文献［15］提出的公平调度器[15]和提出的位置感知调度器。实验结果显示云电子书系统能够提升 73%的 Open VG 类库的性能，减小 50%～75%文件转化过程中的时间消耗，并且提高云平台中 10%的数据命中概率。

3.1 实验环境

将实验环境划分为两个：一个是手持设备中的向量图形类库；另一个是 Hadoop 平台中的向量图形转化器。手持设备环

境中，本文将云电子书安装在一个 Google Nexus 10 平台上。云电子书支持版本为 2.3 及其之后的 Android 系统，因此可以将云电子书安装在大多数的手持设备上。此外，移动端通过网络与云平台进行通信，因此，无线网络频段中的干扰不能对移动端的性能产生影响。

云平台主要由两个部分构成：第一个部分是 Xen 云平台（在多个物理机器上），第二个部分是 Hadoop 集群（在 Xen 集群上）。将这些具有 Xen 云平台的物理机器（服务器）作为 Xen 集群，在 Xen 集群中，每一个物理机器上都运行多个虚拟机（Virtual Machine，VM），管理员也可以通过控制面板管理/安装虚拟机。硬件资源的信息如表 2 所示。

表 2　物理硬件资源

资源	描述
CPU	Intel Xeon E5－2603 Quad－Core
RAM	32GB RDIMM，1333MHz(8GB＊4)
存储器	500GB 7.2K RPM SATA 2.5＊10
网络	Ethernet 10/100 Base－TX
VM	VM 的个数为 24 个
O. S.	Ubuntu 12.04 ＋ XEN 虚拟层
手持设备	Google Nexus 10（Android）

3.2　向量图形类库性能

通过运行多个标准对向量图形类库性能的提高进行测试，这些基准包括 Tiger VG、Apple VG 和 Star VG，实验也对 Open VG 流程中不同阶段的性能进行了测量。表 3～5 给出了不同方法所测量的所有性能值。

Tiger VG 的实验结果如表 3 所示，从表 3 中可以看出向量图形类库在三角划分步骤的的性能明显提升，这是因为在最初的设计中三角划分具有较大的图形计算任务负载，在云电子书中，三角划分的负载可以转嫁给嵌入式 GPU 计算单元。

表 3　Tiger VG 的各阶段执行时间　　ms

阶段	TigerVG－Open VG	TigerVG－AndroidVG	TigerVG－云电子书	TigerVG－云电子书(GPU)
扁平化	2000	2100	2300	1900
转化	10	10	10	10
三角划分	100000	35000	32000	19000
光栅化	8000	7500	6000	4900
整体性能	110010	44610	40310	25810

Apple VG 的实验结果如表 4 所示，Apple VG 的三角划分步骤性能也提升较大，Apple VG 的执行时间减少约 60%。另一方面，虽然 Star VG 的结果如表 5 所示，虽然含有许多较轻权重的计算任务，但性能依然有所提高。

表 4　Apple VG 的各阶段执行时间　　ms

阶段	AppleVG－OpenVG	AppleVG－AndroidVG	AppleVG－云电子书	AppleVG－云电子书(GPU)
扁平化	1000	1170	1030	940
转化	10	10	10	10
三角划分	45000	17000	15000	12000
光栅化	5000	5090	5000	4800
整体性能	51010	46540	21040	17750

表 5　Star VG 的各阶段执行时间　　ms

阶段	AppleVG－OpenVG	AppleVG－AndroidVG	AppleVG－云电子书	AppleVG－云电子书(GPU)
扁平化	180	175	175	175
转化	40	40	40	40
三角划分	500	300	280	260
光栅化	800	740	710	600
整体性能	1520	1225	1205	1075

云电子书在 Star VG 和 Apple VG 的实验中明显的提升这两种实验环境的性能。与之相反的是，Star VG 的性能并没有提升很多，这是因为 Tiger VG 和 Apple VG 是复杂的 2D 动画片。云电子书通过优化算法和嵌入式 GPU 计算器能够减少复杂 2D 动画片的执行时间。另一方面，Star VG 是一个简单的 2D 图形，这个图形不需要多余的计算能力。

3.3　云平台的性能

云平台的实验结果在每个阶段分别解释对增强方法效果的影响。首先，云电子书采用图像群组框架的并行方法，将电子书中所有的位图格式图像划分为多个群组，然后并行地在 Hadoop 平台上对位图格式图像组进行转化。本文将云电子书和其它方法进行了性能增强方面的比较，方法包括具有顺序处理的服务器和常规的 Hadoop 平台。图 2 中给出了提出方法所获取的性能增强结果，以及总的执行时间。图 2 中，将实验任务从 2500 设置为 100000。当任务的个数增加时，云电子书获取了显著的性能提升。当任务的个数达到 100000 时，可获得 55%的性能提升。

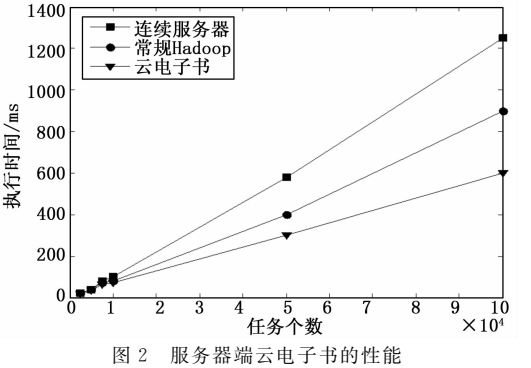

图 2　服务器端云电子书的性能

为了展现本文提出工作调度器的性能，在两个不同的分布中设置了三个不同的负载。实验中，将每分钟处理 15000 个任务设置为高，将每分钟处理 10000 个任务设置为中，将每分钟处理 5000 个任务设置为低。此外，利用一个结合正太分布和泊松分布的仿真客户端安装任务生成器以生成实验负载。在服务器端对所提出的调度器进行了仿真实验。

表 6 给出了当工作跟踪器采用不同调度算法时工作跟踪器中遗漏计数个数（高度负载）。当跟踪器采用本文所提出的算法作为其调度器时，工作跟踪器内遗漏计数的个数可以减少 150～300。表 7 给出了运行中型负载情况下不同调度器的工作跟踪器的遗漏比例。如表 7 所示，本文提出方法对应的数据遗漏比率性能与先进先出（first in first out，FIFO）调度器相比要高出 10%。类似地，如图 8 所示，在较轻负载情况下，提出的方法也可以减少 10%的数据遗漏比率。

（下转第 174 页）

度的情况下，得到植入轮胎的 RFID 标签的读取距离。

表 3 误差分析表

输入值			输出值		误差/dm
介电常数/(F/m)	钢丝层距/mm	植入深度/mm	预测值/dm	实际值/dm	
1.5	3.0	6.0	9.4999	9.5	−8.4239e−05
3.5	3.0	6.0	7.9997	8.0	−0.0003
6.0	3.0	6.0	6.0000	6.0	−1.8763e−05
1.5	5.0	6.0	9.00000	9.0	−9.8313e−06
3.5	5.0	3.0	7.8419	7.8	0.0419

4 结论

BP 神经网络在预测算法中具有很大的优点，首先它可以通过不断地修改权值来增加模型的准确性，其次具有高度的容错性，即使部分神经元损伤也不会破坏整体，网络的高度连接意味着少量的误差可能不会出现严重的后果。然而，神经网络的建立需要大量的数据来进行模型的训练，以确保模型的准确性。因此，通过 FEKO 电磁仿真软件，对标签的长度及植入环境进行了模拟仿真，从而得到神经网络建模所需要的训练数据。通过仿真建模的方式比起实际操作起来更加方便简单，并且具有可行性。对以后不同型号轮胎的长距离信息采集具有一定的帮助作用。

参考文献：

[1] 张志刚. 射频识别 RFID 技术在汽车电子中的应用 [J]. 应用科技，2012，7.

[2] 谢溪凌，汪俊亮，郑卫刚. 飞翔智能轮胎的仿真设计 [J]. 研究与开发，2012，5.

[3] 王曙光，董兰飞，邬立春. RFID 轮胎标签封装设备的设计开发 [J]. 橡胶技术与装备，2010，36 (6).

[4] 周开利，康耀红. 神经网络模型及其 MATLAB 仿真程序设计 [M]. 北京：清华大学出版社，2013.

[5] 王曙光，董兰飞，邬立春. RFID 轮胎标签封装设备的设计开发 [J]. 橡胶技术与装备. 2010，36 (6)：40－42.

[6] 周 品. MATLAB 神经网络设计与应用 [M]. 北京：清华大学出版社，2013.

[7] 郭 涤，周 军. 基于 Matlab 的神经网络预测模型研究 [J]. 物流科技，2006 (1)：125－128.

(上接第 165 页)

表 6 服务器端调度算法的数据遗漏比率(高度负载) %

分布	具有 FIFO 云电子书	具有 LaSA 的云电子书
正太分布	86.2	67.7
泊松分布	89.4	72.0

表 7 服务器端调度算法的数据遗漏比率(中度负载) %

分布	具有 FIFO 云电子书	具有 LaSA 的云电子书
正太分布	26.7	16.1
泊松分布	27.2	14.2

表 8 服务器端调度算法的数据遗漏比率(轻度负载) %

分布	具有 FIFO 云电子书	具有 LaSA 的云电子书
正太分布	15.8	3.0
泊松分布	18.1	6.1

4 结论

本文提出了一种综合电子书系统，即 Android 系统上的云电子书系统，该系统可提供文件转化、数据归档和向量图像表示功能。为了有效地优化云电子书系统，提出了一种应用于云电子书系统的多级别优化框架。在图形描述、计算性能等多个方面进行了优化。与其他几种系统和类库相比，明显提升了整体性能。

由于许多编程人员利用 GPU 计算的优势来提高嵌入式环境中系统类库的性能，而 Open VG 是一种性能较差的嵌入平台实现 2D 向量图形标准接口，后期针对特定平台考虑用 GPU 对其进行改善。

参考文献：

[1] Larson L C. Digital readers: the next chapter in E－Book reading and response [J]. Reading Teacher, 2010, 64 (1): 15－22.

[2] 毛香英，郁 梅，蒋刚毅，等. 基于结构失真分析的立体图像质量客观评价模型 [J]. 计算机辅助设计与图形学学报，2012，24 (8)：1047－1056.

[3] Goya C. Visualization of collecting locations based on scalable vector graphics for the specimen information software [J]. Advanced Materials Research, 2014, 642 (4): 322－331.

[4] 乐银煌. 基于学习型电子书的移动学习模式研究及其应用 [D]. 中国科学技术大学，2014.

[5] 杨志刚，张新兴，庞弘燊. 电子书阅读器在国外图书馆的应用现状及存在问题 [J]. 大学图书馆学报，2011，29 (4)：11－17.

[6] 焦灵芝. 亚马逊电子书平台研究 [D]. 南京：南京大学，2013.

[7] 王 芳. 电子书标准化建设的目的、现状与对策探讨 [J]. 情报科学，2011，27 (6)：953－956.

[8] Sans V, Diaz J. Implementing a multimedia application on iphone: a case study [A]. 2011 IEEE 14th International Conference on Computational Science and Engineering (CSE), 2011: 233－238.

[9] 丁建飞. 基于语义的电子书交互阅读 [D]. 北京：北京交通大学，2013.

[10] Kim D, Cha K, Chae S I. A high－performance Open VG accelerator with dual－scanline filling rendering [J]. IEEE Transactions on Consumer Electronics, 2008, 54 (3): 1303－1311.

[11] Kim D, Cha K, Soo－Ik C. Adaptive Scanline Filling Algorithm for Open VG 2D Vector Graphics Accelerator. [J]. Ieice Transactions on Information & Systems, 2009, 92 (7): 1500－1502.

[12] Jackson W. WatchFaces Vector Design: Using Vector Graphics for WatchFaces [M]. Pro Android Wearables. Apress, 2015.

[13] 杨 瑛，刘文文，吴方贵. 基于 GPU 的可视化测量仪器软件设计 [J]. 计算机测量与控制，2016，24 (8)：150－153.

[14] 姚卫国，张东波. 基于 Hadoop 分布式平台的 Web 文本关键词提取方案 [J]. 湘潭大学自然科学学报，2016，38 (2)：79－83.

[15] Liu J, Wu T, Lin M W, et al. An Efficient Job Scheduling for MapReduce Clusters [J]. International Journal of Future Generation Communication & Networking, 2015, 34 (6): 109－115.