

硕博连读生转博资格报告

网络内存客户端相关研究

报告人：唐 欢

指导教师：樊建平

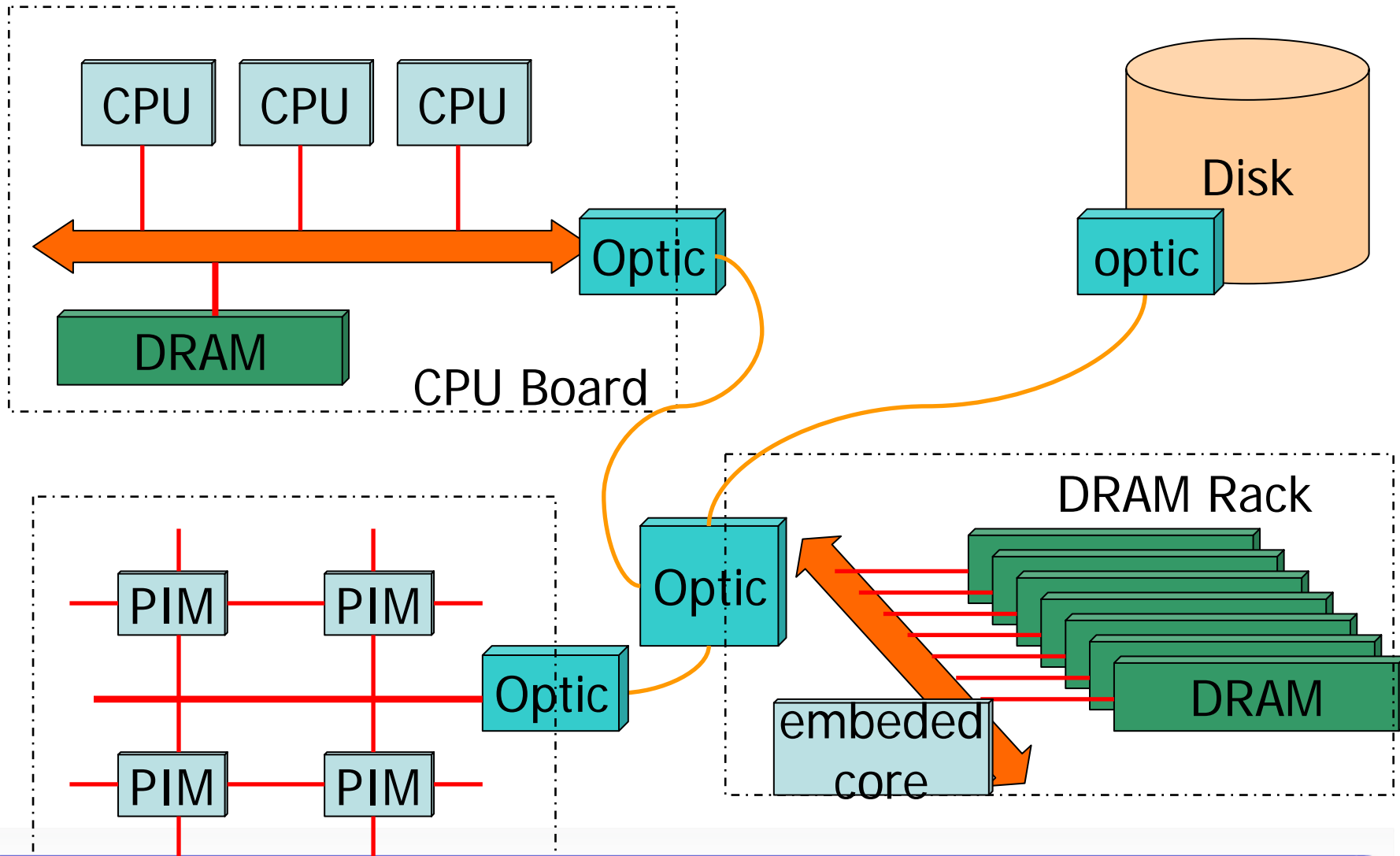
报告日期：2005.06.28

主要内容

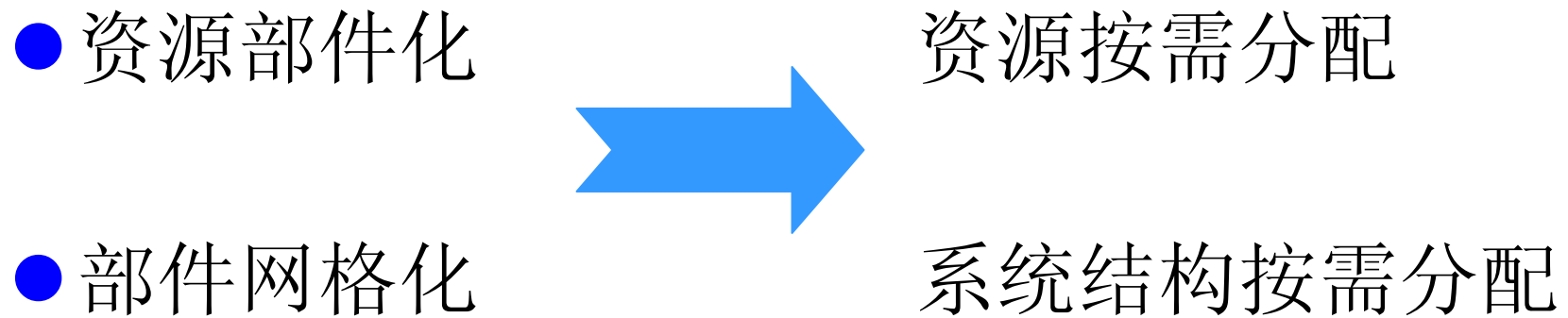
- DSAG体系结构 —— 背景
- 部件化网络内存服务系统 —— 需求
- Memory Load Manager设计实现
- Page Fault Tracker设计实现
- 总结与展望

DSAG体系结构——背景

DSAG体系结构

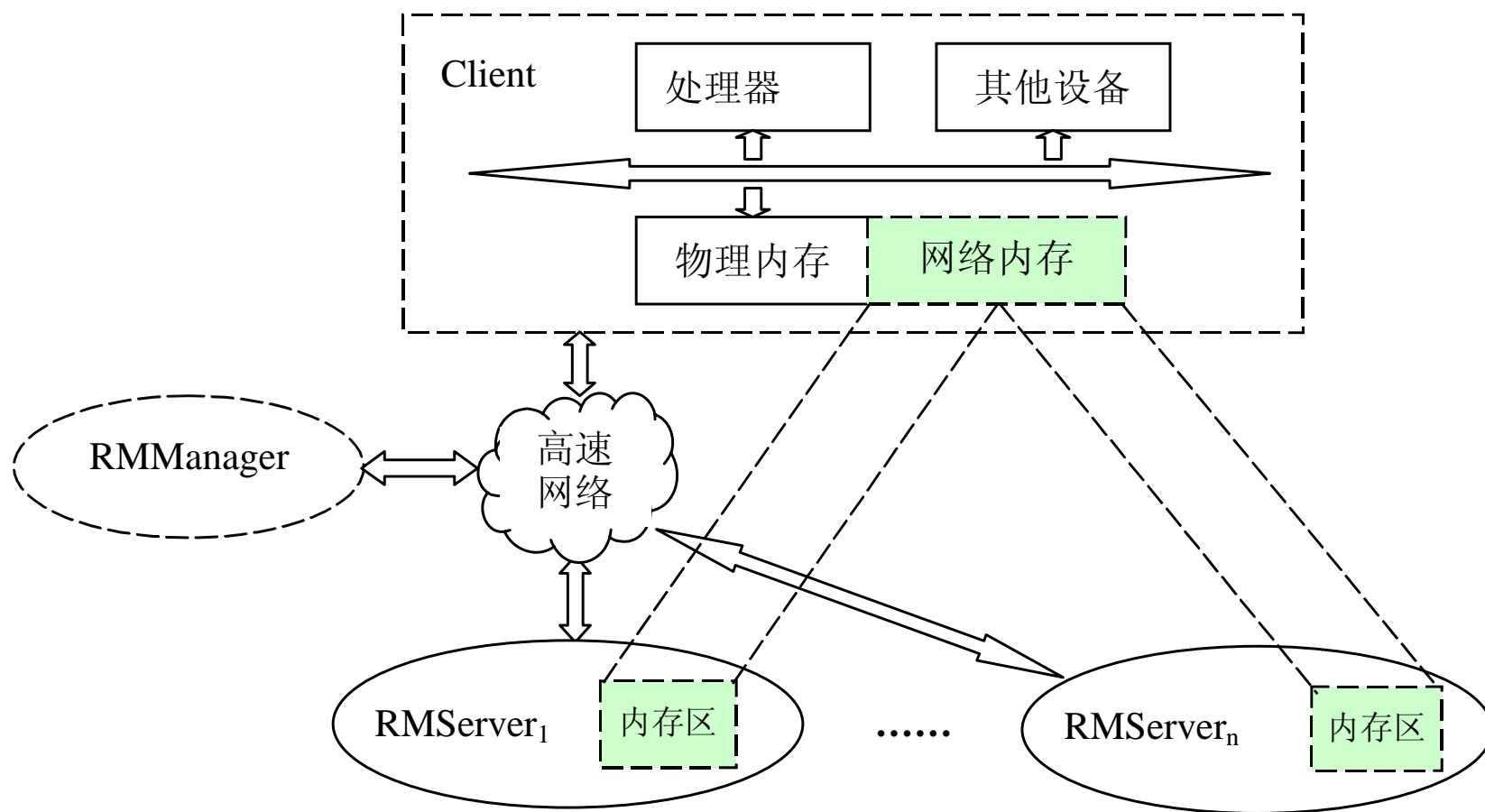


网格化的动态自组织体系结构




部件化网络内存服务系统

系统结构示意图



客户端用户接口

- 标准的块设备接口
- 使用方式：
 - 交换设备方式；
 - 文件系统方式；
 - 用户编程接口。通过mmap系统调用访问网络内存块设备。这是一种不透明的方式。

优点

- **提高资源的利用率。**通过网络内存技术，把原有分散的内存资源集中管理，通过动态共享实现了资源的时空上的交叉利用。以此避免部分节点内存资源大量空闲而同时其他节点内存资源紧张的情况，提高内存资源利用率。
- **降低客户端资源需求，实现低成本信息化。**由于内存资源的集中管理，客户端内存资源配置将降低，同时相同部件集中管理，使得故障率降低、维护和升级成本降低，总体上可以实现全系统的低成本。

问题

- **应用程序性能及效率保证。** 在使用远端内存的情况下，试图减少客户端内存资源的配置量，有可能会影响应用程序的性能及执行效率。严重的可能会出现全系统的“瘫痪”现象。
- **远程内存访问延迟。** 由于通过网络对远端内存进行访问，较原有通过总线方式访问，有所延迟。

解决方法

- 应用程序性能效率保证 → 客户端内存资源控制
- 远程内存访问延迟 → 延迟隐藏

我的工作

- 客户端内存资源控制
 - ➔ Memory Load Manager
- 延迟隐藏
 - ➔ 内存调度策略、cache管理策略、预取
 - ➔ 应用程序访存特性研究
 - ➔ Page Fault Tracker

内存资源控制的设计实现

——Memory Load Manager (MLM)

- 系统概述
- Linux内存页面回收流程
- MLM 设计实现
- MLM办公应用测试
- 后续工作

MLM概述

- 目标

- 对应用程序使用物理内存资源的限制

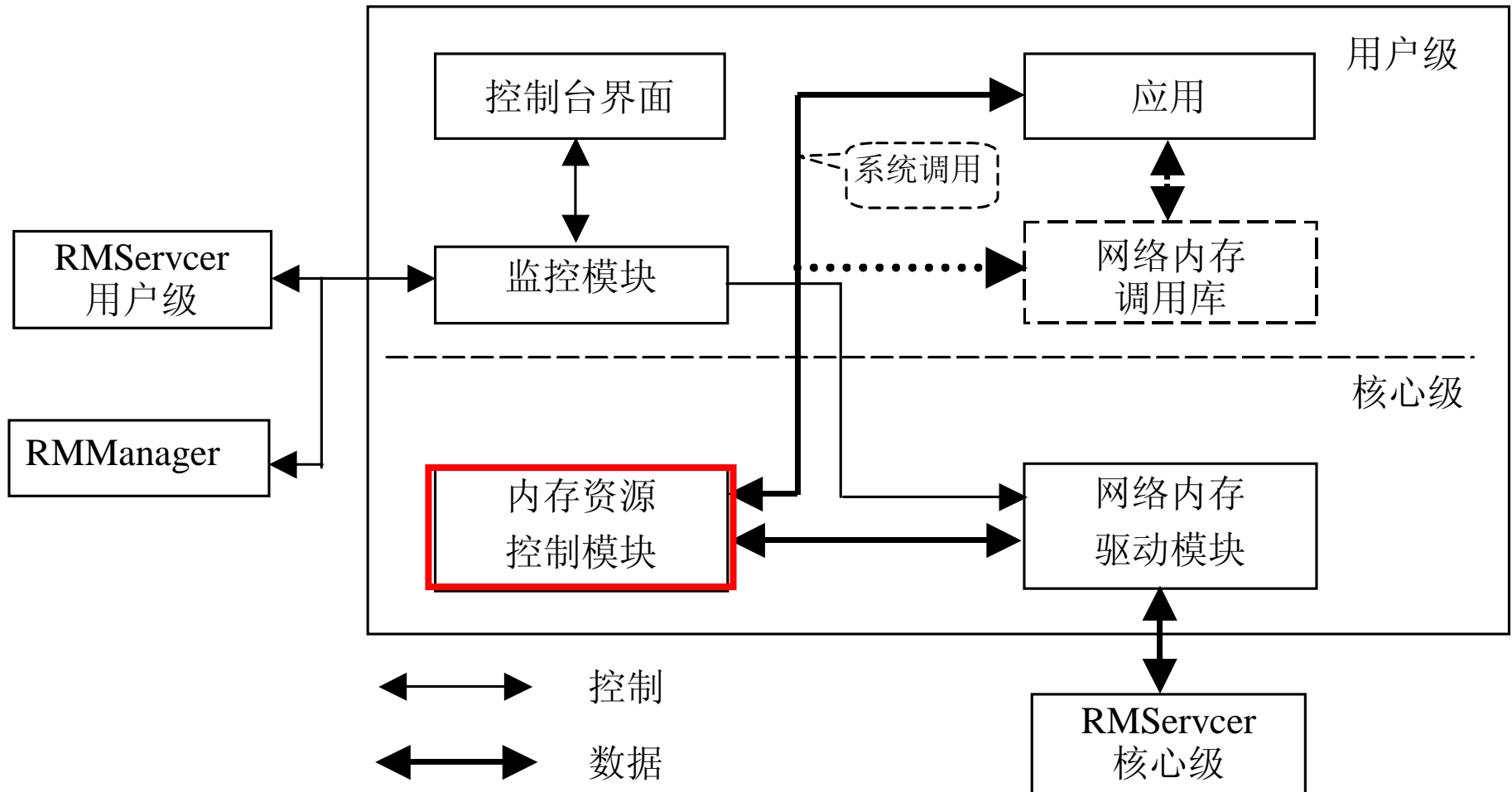
- 实现手段

- 修改Linux内核 (2.4.28版本)

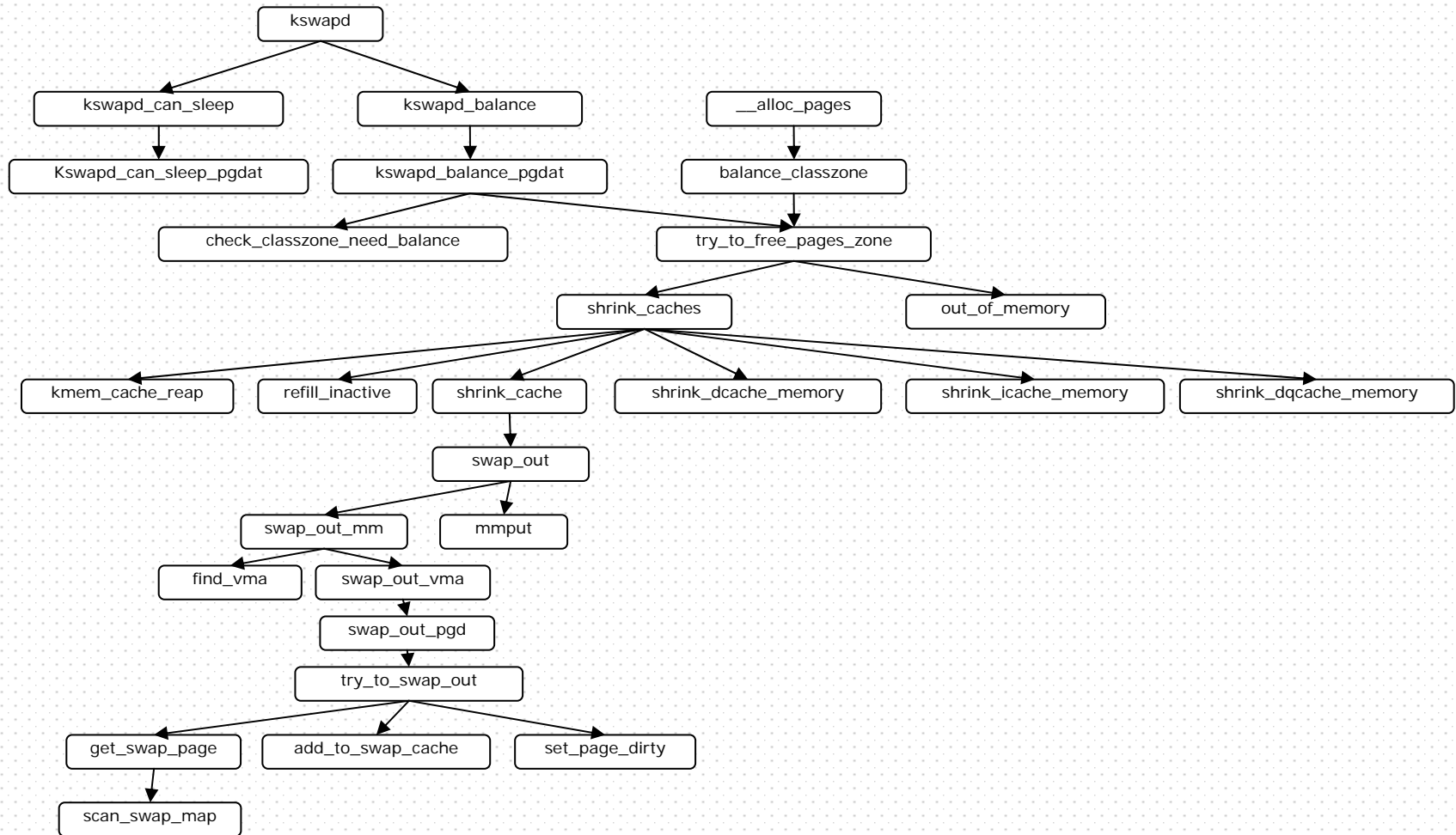
- 实现方法

- 对应用程序规定内存使用量限制值
- 超出限制值的部分被换出(swap out)

网络内存系统客户端软件结构



Linux内存页面回收流程



Linux内存页面的回收时机

- 内核线程kswapd周期性的被唤醒，检查所有内存区域(zone)的内存使用压力，对于内存使用量过度的内存区域进行内存页面的回收处理。
- 当申请新的内存页面请求无法被满足时，对页面所在的内存区域进行内存页面的回收处理。

Linux内存页面的回收对象

- 页面回收过程是否进行，由**全系统**内存使用量是否紧张决定；
- 在选择待换出页面时，系统对所有页面平等看待，也即任何**进程的页面都有平等的机会**被换出，内核并不对页面所属的进程区别对待；

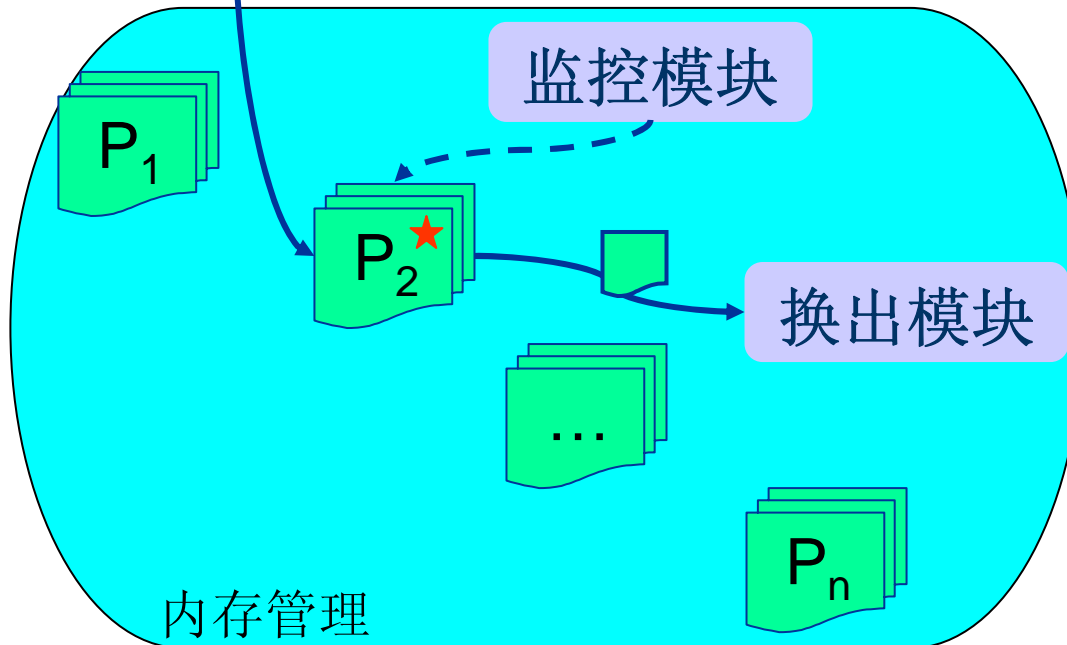
Memory Load Manager 结构

系统调用接口

用户级

核心级

监控模块



换出模块

内存管理

Net Ram

MLM功能描述

- 可对任何进程进行资源限制，使得系统中所有进程都具有物理内存资源使用量的**限制值**；
- 提供对进程物理内存资源使用量的**监控模块**，当使用量超过其限制值时，引发交换；
- 提供对**单一进程地址空间**的扫描并交换的功能模块，使得系统仅处理特定页面，这些页面属于超过内存资源限制值的进程；

Memory Load Manager

办公应用测试

试验目的

- 为了测试同等环境下网络内存(NetRam)与本地磁盘相比，对于办公应用程序性能的影响。
- 预期证明使用网络内存可以在保证应用程序性能的情况下，降低本地系统内存资源的配置。

实验环境

➤ 客户端

- CPU : PentiumIV 2.8G
- Mem: 64M
- Disk : scsi
- OS : Redhat linux-2.4.28-mlm01 (定制内核)
 - ◆ mlm: Memory Load Manager 。

➤ 服务器

- CPU : AMD Opteron 1.6G smp
- Mem: 2G
- OS : Redhat linux-2.4.27

试验方法

➤ 网络内存使用方式

- 交换设备

➤ 应用程序

- Open Office Impress (PPT文件)
- Adobe Acrobat (PDF文件)
- Gimp (图像处理)

➤ 记录数据

- 各应用程序在不同内存限制值的情况中，运行相同动作序列所需要的时间。
注：动作序列由GUITest脚本自动发出。

试验方法 – 续

● 对比实验

- 本地磁盘(HardDisk)作交换设备
- 网络内存(NetRam)作为交换设备
- 本地内存(RamDisk)作为交换设备

使用Ramdisk作为交换设备时的性能表现是交换设备能够带来的理想情况下的最佳表现，RamDisk的性能将是NetRam性能的上限。

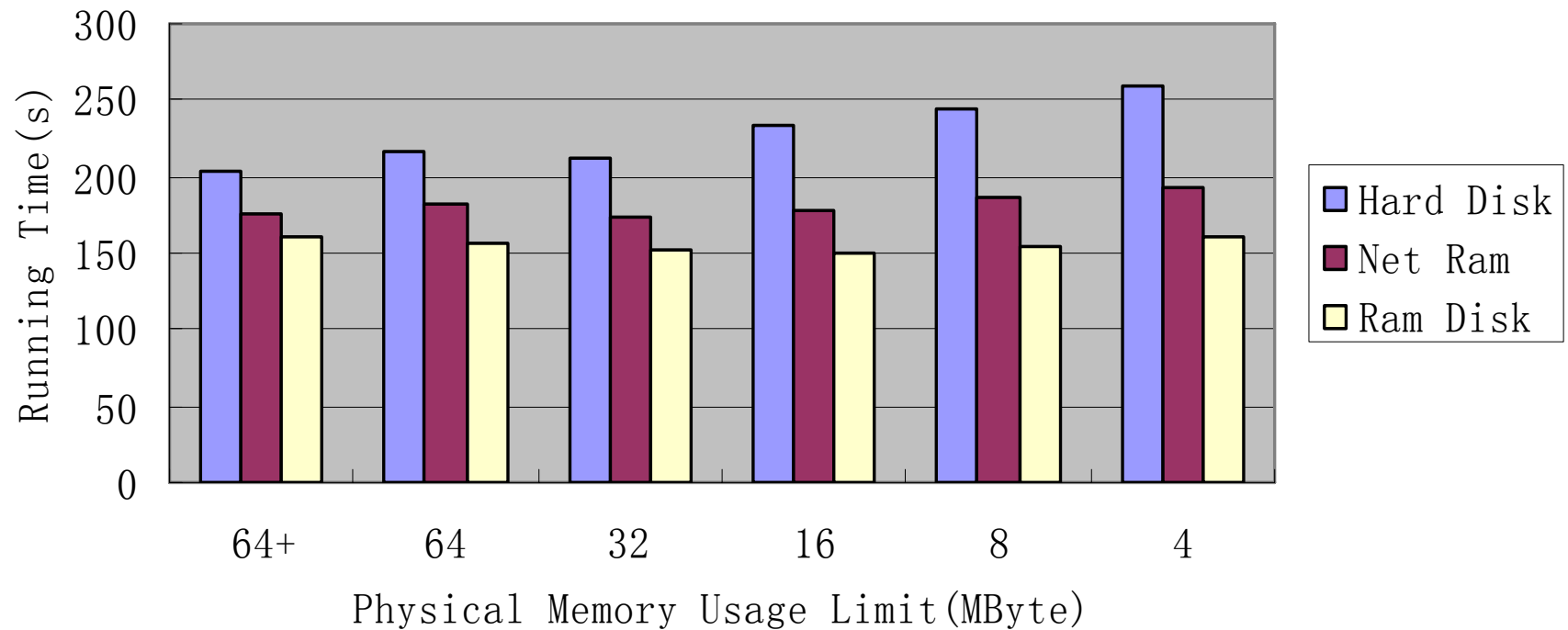
Open Office Impress

- 动作序列

- 启动Open Office Impress;
- 打开 PowerPoint 文件(21 Mbytes);
- 全屏幕播放 51 页;
- 退出

Open Office Impress

Runing Time of PPT Using Different Swap



64+: 64MB local physical memory , NO limit

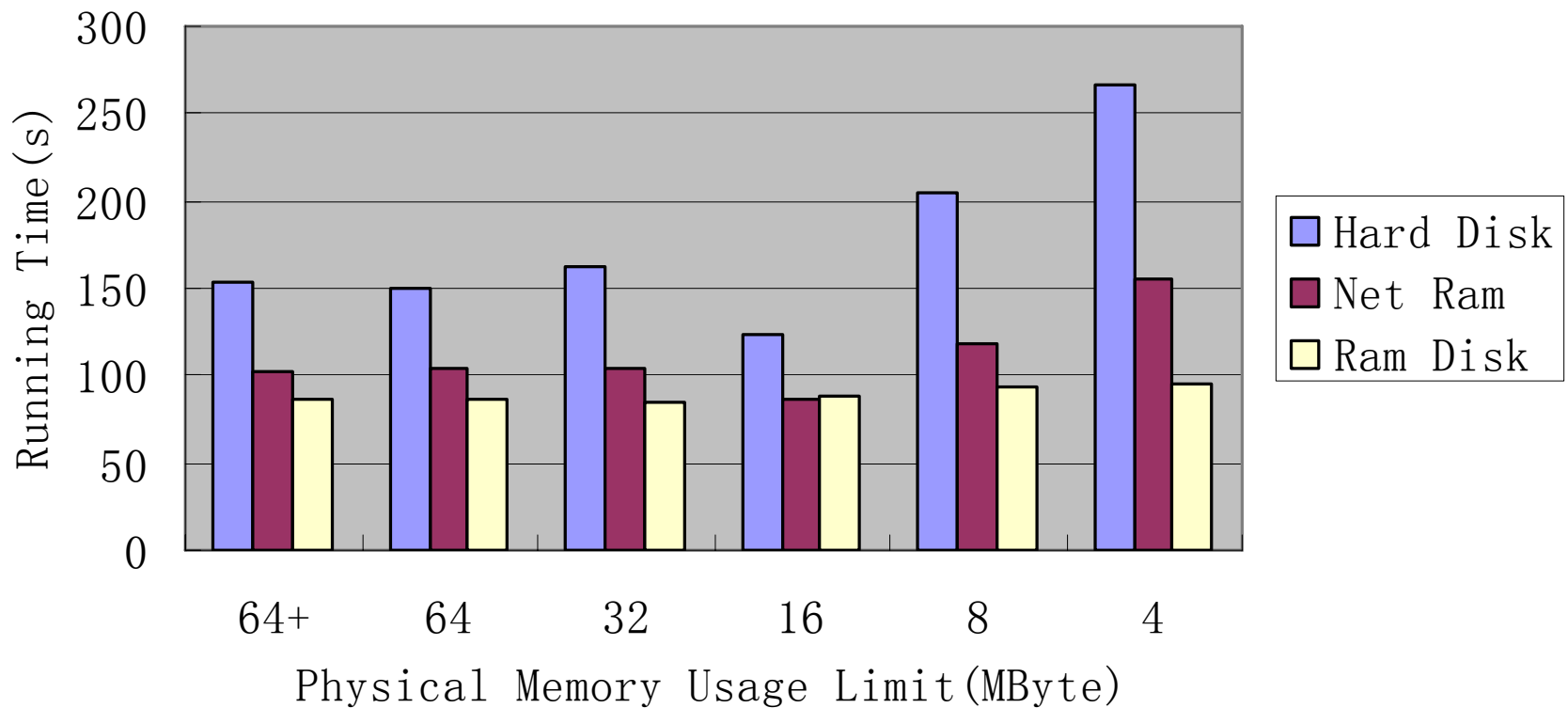
Adobe Acrobat

● 动作序列

- 启动 Acrobat;
- 打开 pdf 文件 (85 Mbytes);
- 随机翻阅其中 50 页 (伪随机);
- 退出

Adobe Acrobat

Runing Time of Acrobat Using Different Swap



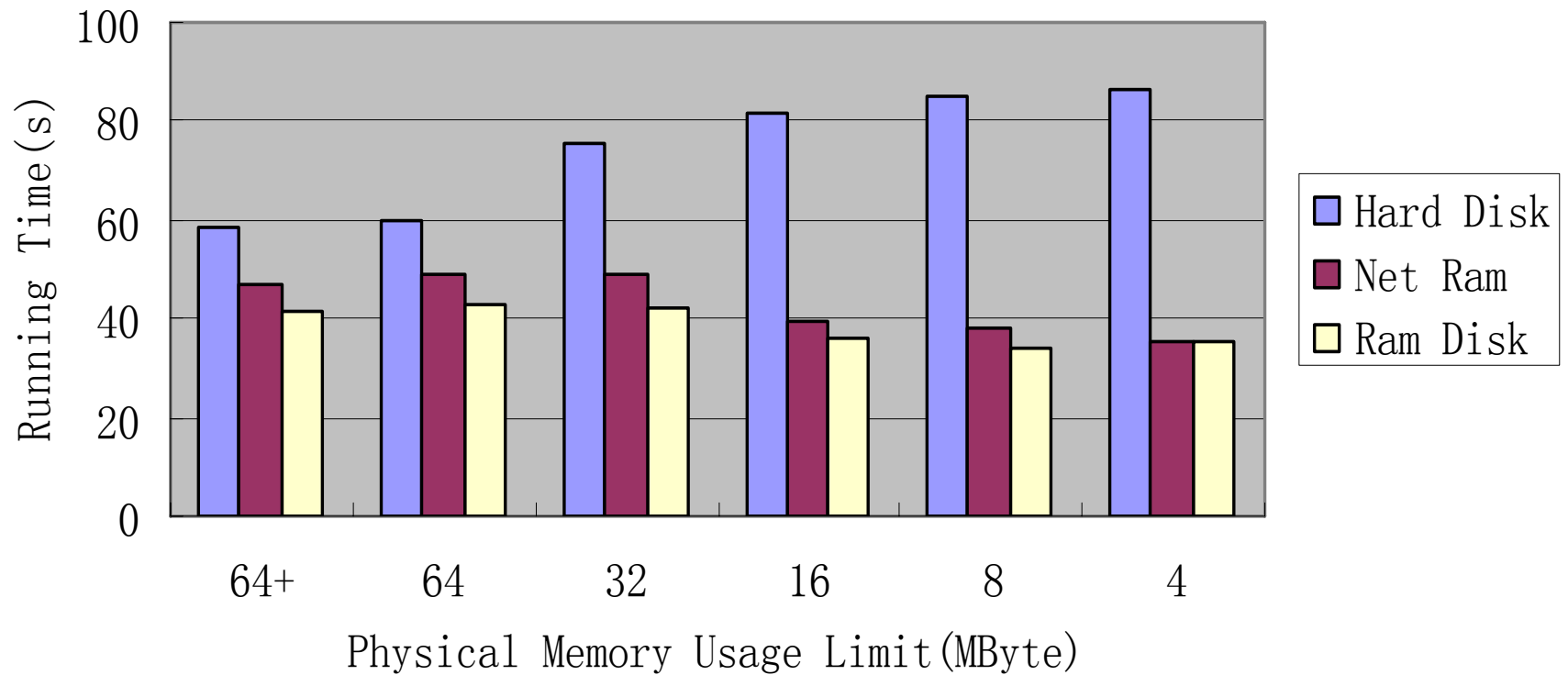
Gimp

● 动作序列

- 启动 Gaim;
- 打开图像文件(4.1 Mbytes 巴格达卫星图);
- 将图片放大到 1:1 视图;
- 图像窗口最大化;
- 图像180° 反转;
- 退出.

Gimp

Runing Time of Gimp Using Different Swap



结果分析

- 使用网络内存作为交换设备的情况下，应用程序执行效率高于使用传统磁盘的情况，并且接近使用RamDisk的情况。
- 在对应用程序进行限制时，由于其无用页面被及时换出，为系统提供了更多的内存资源，反而可以使得应用程序性能提升。
例如：Gimp图像处理中NetRam的性能表现，在内存使用限制达到32Mbytes时，性能有所提升。

结果分析 – 续

- Memory Load Manager 对应用程序的内存资源限制并没有过多的影响到应用程序执行效率。
- 应用程序执行效率在以下模式下相当
 - 小内存 + Net Ram + MLM
 - 大内存 + Hard Disk

结论

- 网络内存作为交换设备，较传统磁盘的性能有所提高。
- Memory Load Manager与网络内存相结合的模式，保证了小内存情况下应用程序执行效率。
- 在办公应用系统中，可以在一定程度上降低客户端内存配置，并通过网络内存实现资源共享，从而在总体上降低成本。

MLM后续工作

● 内存管理策略

➤ 目前问题:

- 换出页面选择采用进程地址空间轮循 → “颠簸”现象
- 仍然使用系统 swap cache

➤ 后续方向:

- 多种换出策略
- 内存调度策略
- cache管理策略
- 页面预取

适合应用特性
充分利用网络内存特性
避免“颠簸”
延迟隐藏

MLM后续工作 – 续

● Linux面向进程(组)的资源管理

➤ 已有工作:

- AIX Work Load Manager (physical memory)
- Solaris Resource Manager (virtual memory)
- Red Hat Linux 9.0 (Kernel 2.4.20-8)

➤ 后续工作:

- 借鉴2.6内核内存管理(反向映射)
- Open Solaris
- 分组的资源管理
- 资源管理引入优先级

面向进程(组)的资源管理

Page Fault Tracker (PFT)

设计实现

- 系统概述
- Linux Page Fault 简介
- PFT 设计实现
- PFT 测试
- 后续工作

PFT概述

- 目标

- 研究应用程序访存(交换)特性

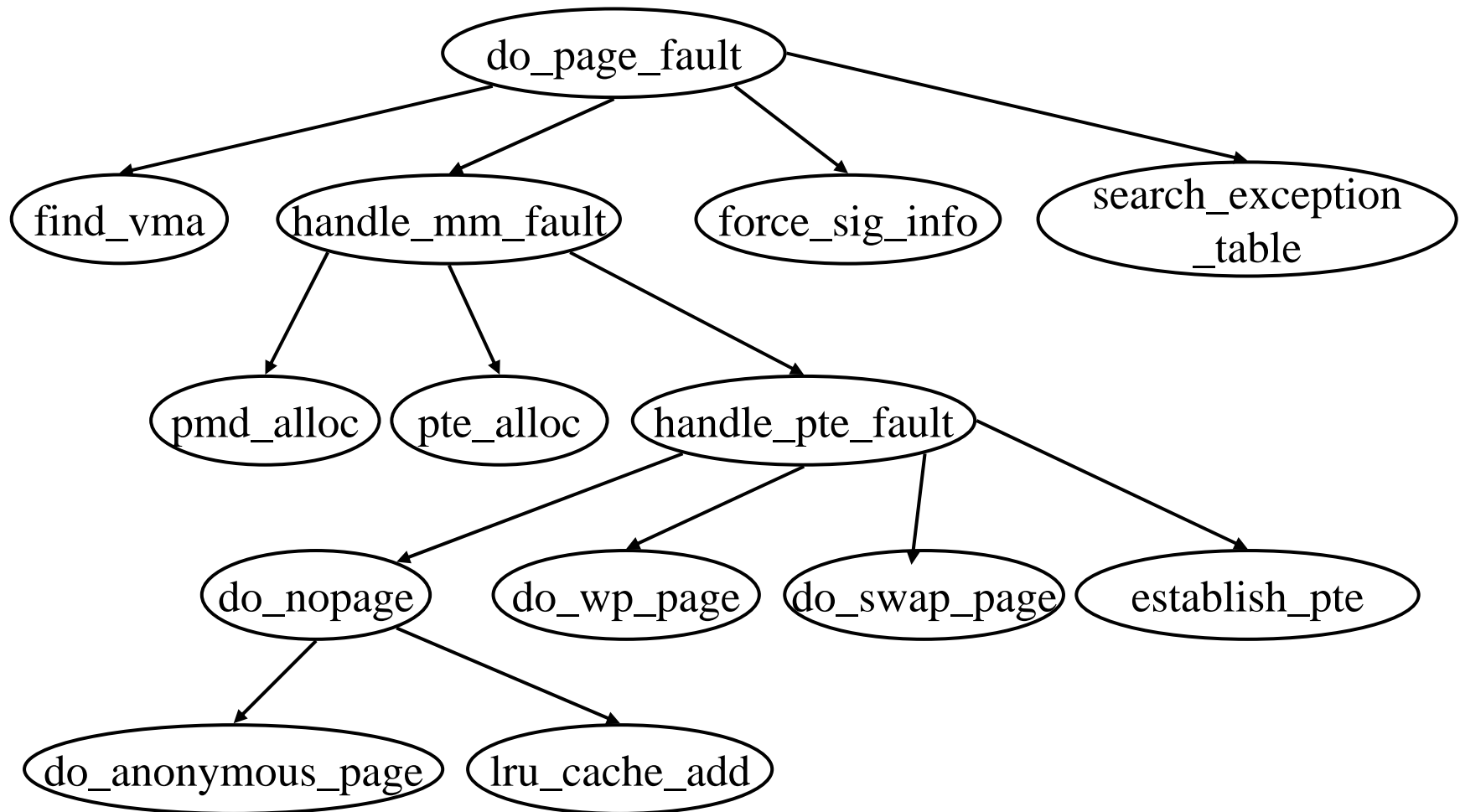
- 实现手段

- 修改Linux内核 (2.4.28版本)

- 实现方法

- 截获指定进程缺页中断产生地址及类型

Linux Page Fault 异常处理



引起Page Fault的原因

<i>Exception</i>	<i>type</i>	<i>Action</i>
进程内存区有效，但没有分配页	minor	分配一个新的物理页
进程内存区a无效，但是在一个可扩展内存区b一侧	minor	扩展内存区b
页面被交换出去，但是位于swap cache中	minor	把页面从swap cache中移除，并分配给进程
页面被交换出去到交换分区	major	从PTE中定位页面位置，从交换分区读入
对标志成只读的页进行写操作	minor	如果页面是COW页，则拷贝生成新的页面，标志成可写
进程内存区有效，但进程没有权限操作	error	例如无效写操作，向进程发送SIGSEGV信号
发生在核心地址空间的异常	minor	发生在Vmalloc区的异常，重新分配一个新的页面
核心态下访问用户内存区异常	error	意味着核心与用户空间交互错误，导致页异常，是核心错误

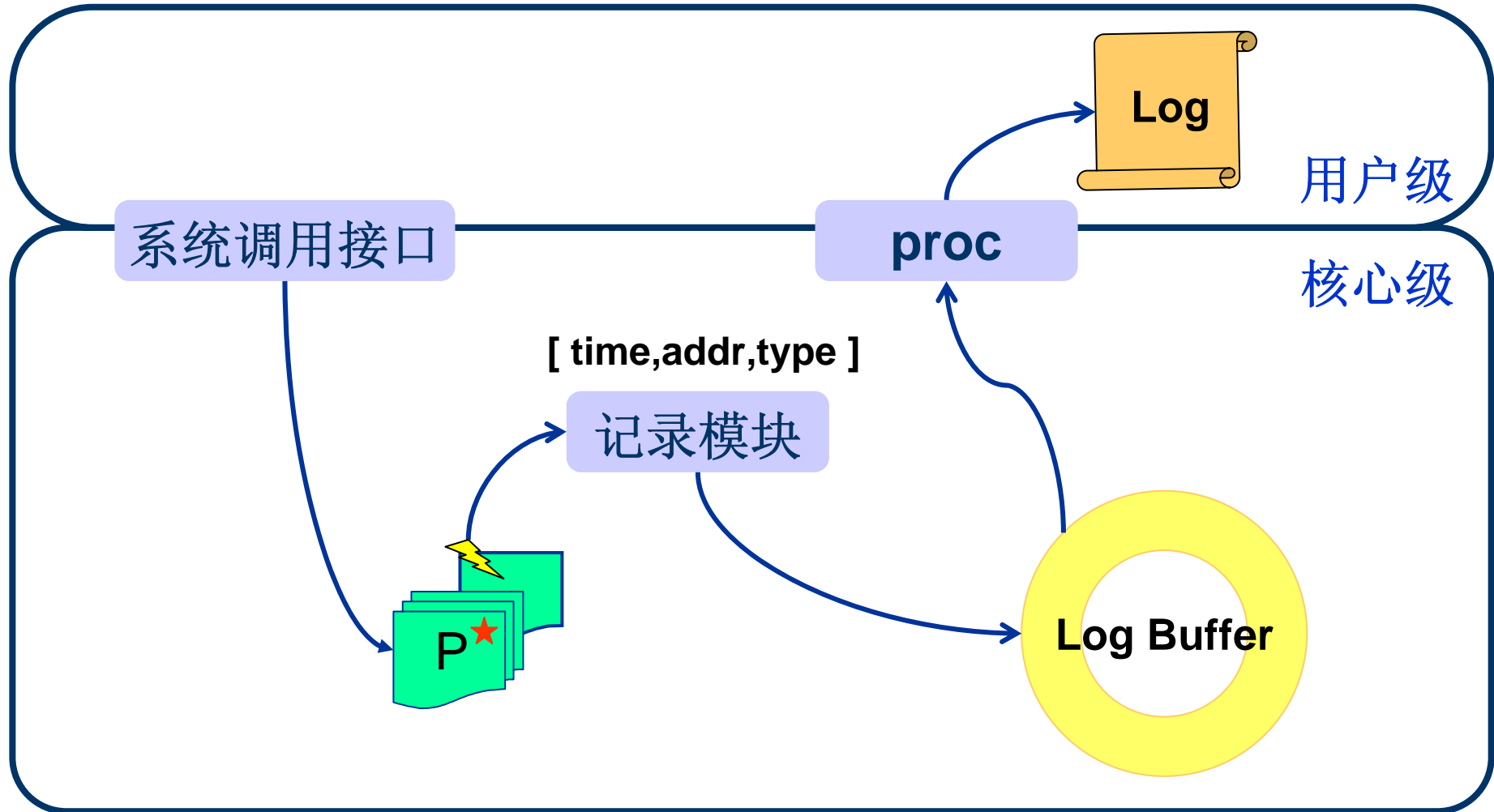
Page Fault 类型

非磁盘访问
Minor

磁盘访问
Major

swap in

Page Fault Tracker 结构



Page Fault Tracker

测试结果及分析

试验目的

- 为了测试不同应用程序运行过程中发生缺页异常时的地址及缺页原因
- 试图借此了解应用程序访存特性，特别是交换(swap)特性

实验环境

- CPU : PentiumIV 2.8G
- Mem : 64M
- Disk : scsi
- OS : Redhat linux-2.4.28-pft01 (定制内核)
 - pft: Page Fault Tracker

试验方法

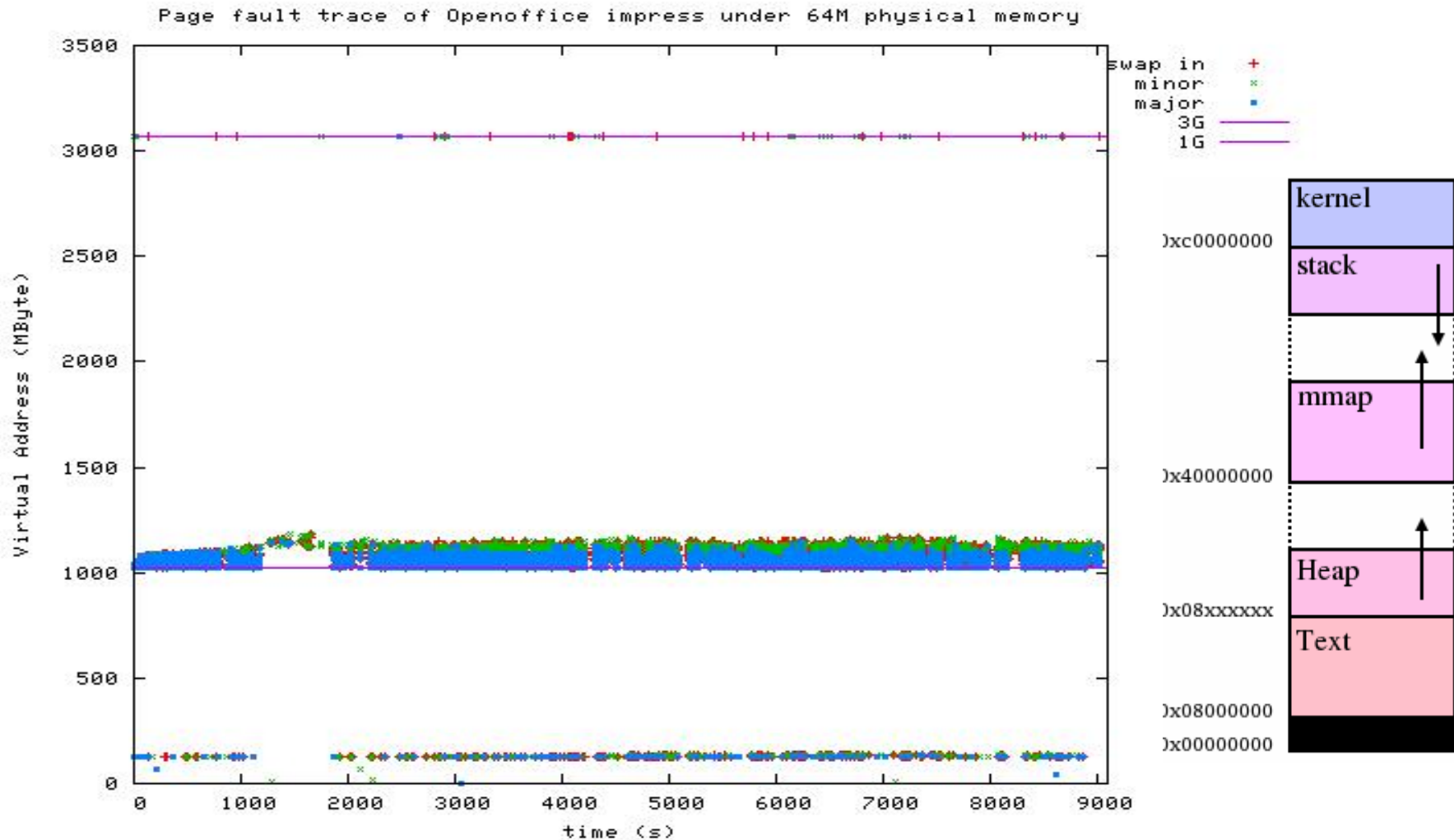
➤ 应用程序

- Open Office Impress (PPT文件)
- Mozilla 网页浏览器
- Eyes Of Gnome 图像浏览
- Vi 文本编辑

➤ 记录数据

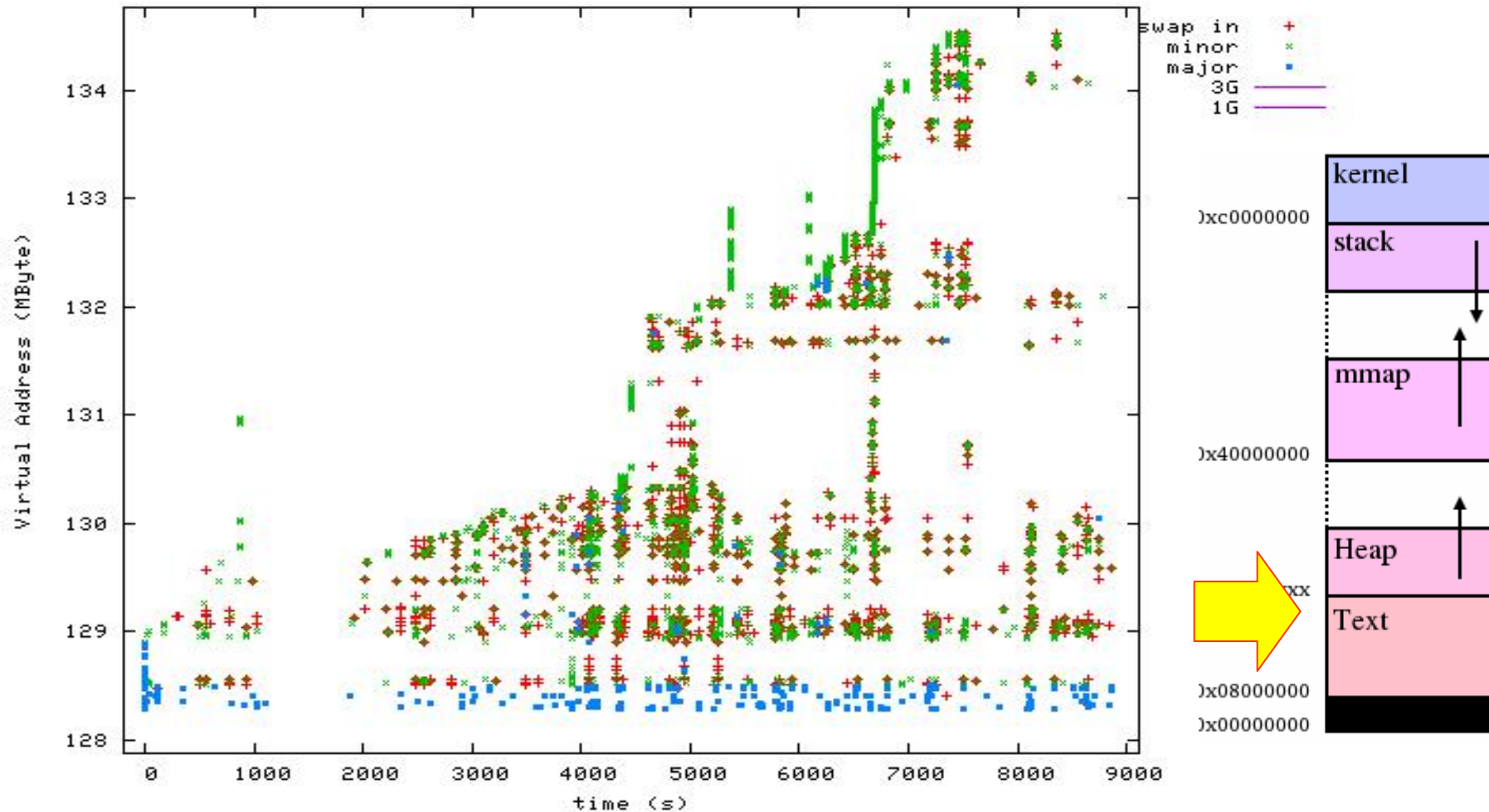
- 应用程序生命周期中发生Page Fault 异常的
[时间, 异常发生地址, 异常类型]

Open Office Impress



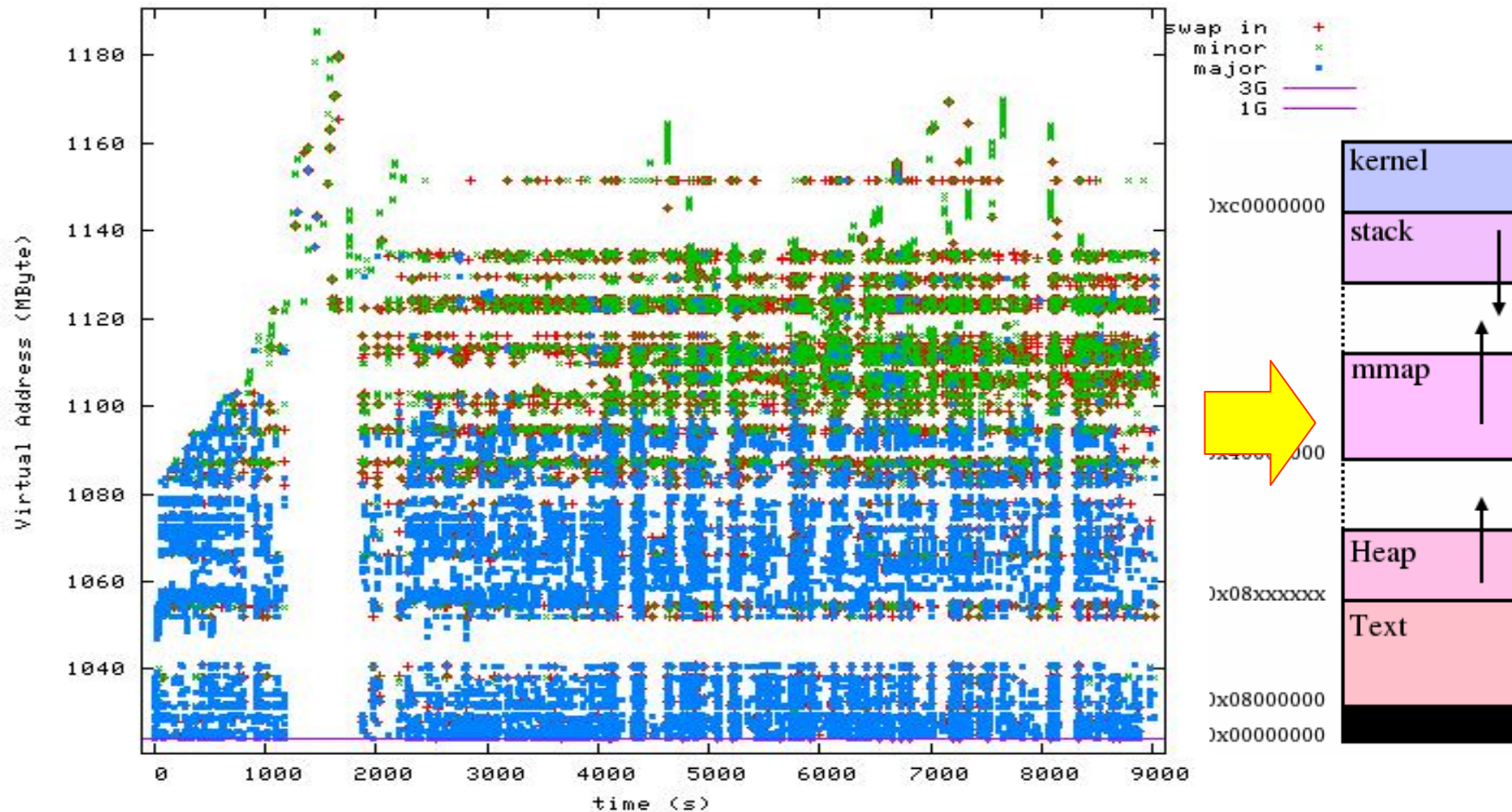
Address: 0~1G

Pagefault trace of OpenOffice Impress under 64M physical memory
Low address



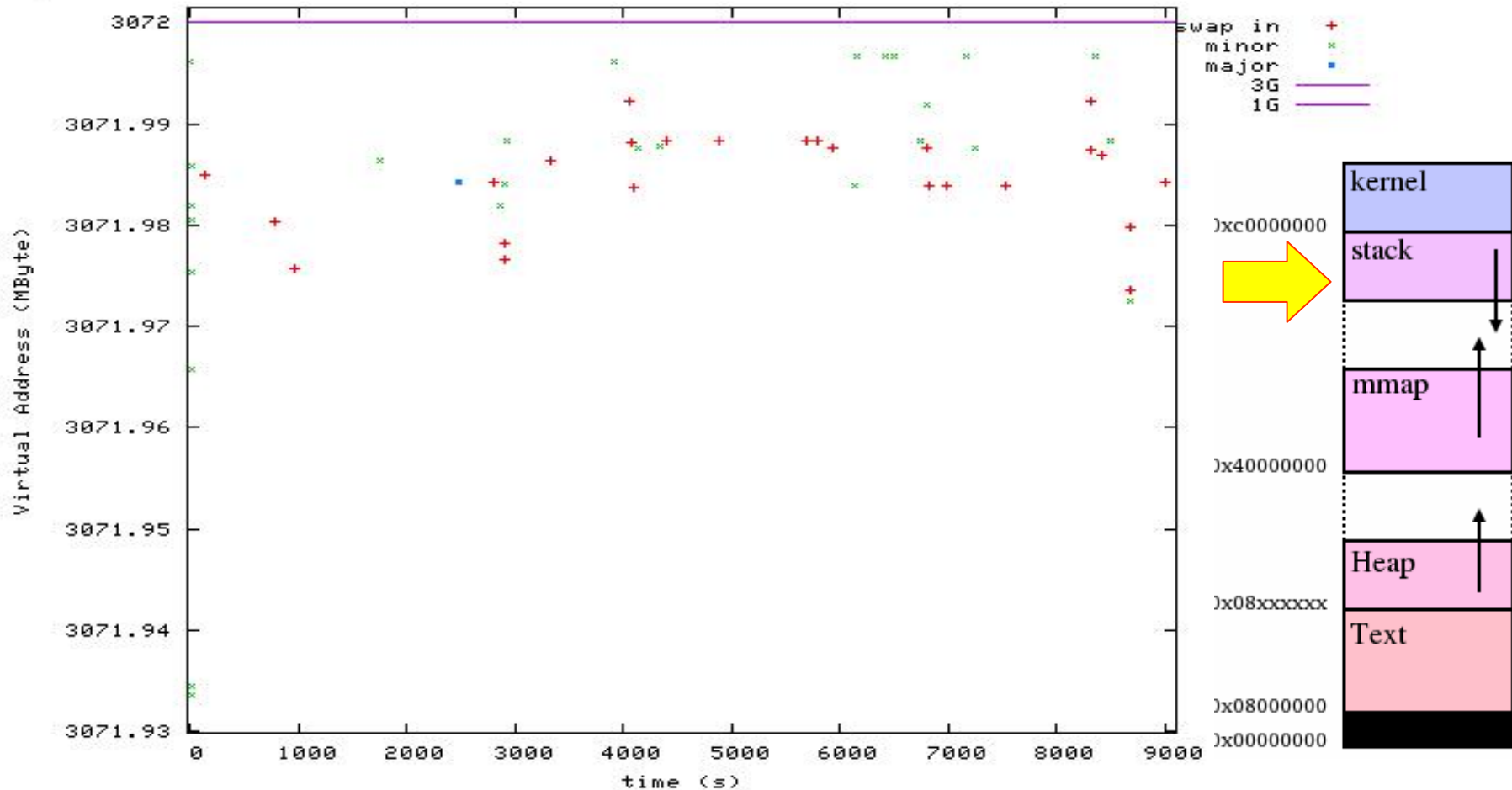
Address: 1G +

Pagefault trace of OpenOffice Impress under 64M physical memory above 1G



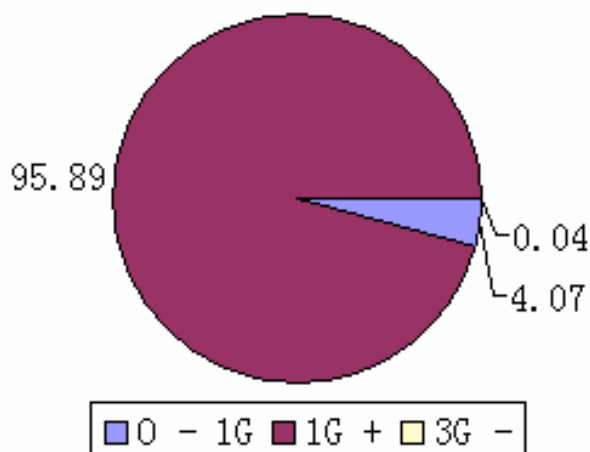
Address: 3G—

Pagefault trace of OpenOffice Impress under 64M physical memory
below 3G

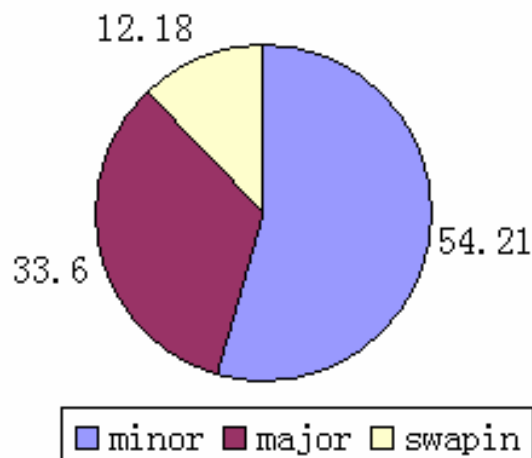


OOImpress 结果统计

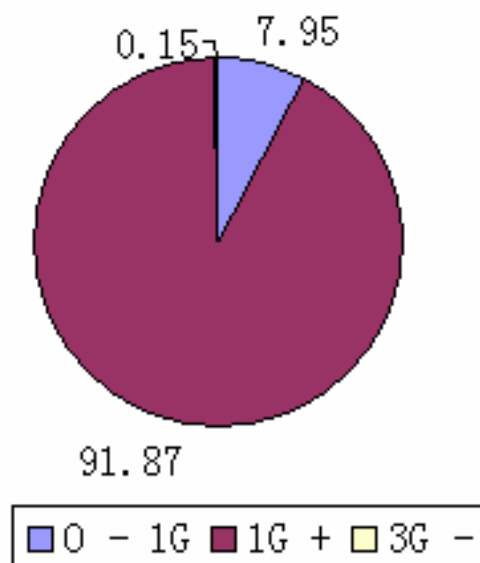
Trace三区间分布



异常类型比例



swap in 区间分布



结论

- 根据page fault 发生地址的分布可推断：进程访存行为在空间也呈现“三段式”分布。(已验证)
- 交换多发生在进程mmap区间，应该作为今后研究的主要关注对象。

PFT后续工作

- 与Memory Load Manager 结合
 - 测试不同内存限制环境中应用程序交换特性
 - 测试不同应用程序的交换特性
- 内存管理策略研究
 - 内存管理策略研究: Trace → 行为 → 策略
 - 策略的验证与评价: 策略 → Trace → 评价

总结与展望

一句话简答

➤ 做了什么？

- Memory Load Manager (MLM)
- Page Fault Tracker (PFT)

➤ 为什么做？意义何在？

- 有效使用现有网络内存系统
- 研究访存特性、实现适用于网络内存环境的内存管理策略

➤ 如何做？

-  修改内核 → 测试 → 评价

➤ 评价标准？

- 应用程序性能

个人今后展望 - 软件vs.硬件



谢谢！