软件架构设计（读书笔记）

刘涛
中科院计算所

版本：0.1

更新：June 22, 2019

——大型网站技术架构与业务架构融合之道

什么是架构

定义 0.1 (架构) 架构是针对所有重要问题做出的重要决策。

本书目的：**借业务架构和技术架构的融合，来建立一种系统化的思维方式和学习方法。**

1 架构的分类

自底向上：

1. 基础架构
2. 中间件与大数据平台
3. 业务系统架构
   - 通用软件架构
   - 离线业务系统架构
   - 在线业务系统架构（本书聚焦）

2 架构的道与术

知行合一。

表 1: 道与术的辨证关系

术	道
外部招式	内功心法
行（实践）	知（理论）
答案	问题

3 语言

俗话说：千招会不如一招熟。

精通一门语言。

现代高级编程语言的共同特征：

- 基本数据类型
- 类型转换、类型推断、类型安全
- 顺序、选择、循环三种语句类型
- 面向对象：封装、继承、多态
- 常用数据结构的库
- I/O 库
- 线程库（或者协程库）

注 学习 C++，需要懂对象的内存布局、编译器和连接器内部如何工作……

4 操作系统

4.1 缓冲 I/O 和直接 I/O

几个关键概念：

定义 4.1 (应用程序内存) 我们开发的代码中用 *malloc/free*（或 *new/delete*）分配出来的内存空间。

定义 4.2 (用户缓冲区) C 语言 *FILE* 结构体中的 *buffer*。作用是：减少系统调用次数，降低 *I/O* 过程中操作系统在用户态和内核态之间切换的消耗。

定义 4.3 (内核缓冲区) *Linux* 操作系统的 *Page Cache*。作用是：**在操作系统层次提高磁盘 *I/O* 效率，优化磁盘写操作。**

定义 4.4 (缓冲 I/O) *C* 语言提供的库函数（以 *f* 打头）：*fopen*, *fclose*, *fseek*, *fflush*, *fread*, *fwrite*, *fprintf*, *fscanf*……

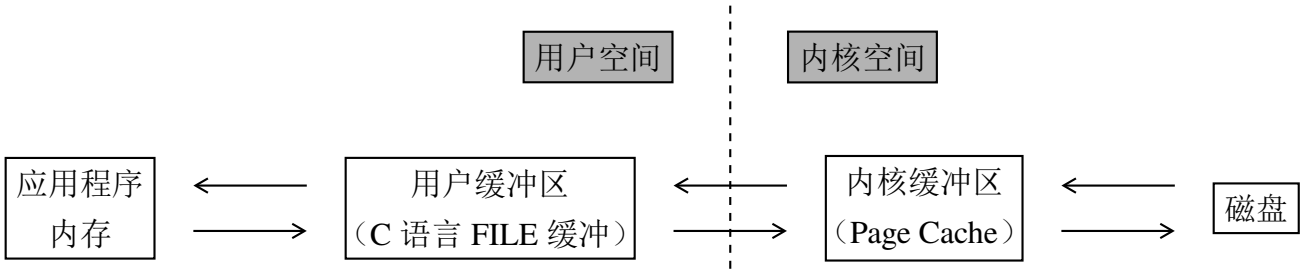

图 1: 缓冲 I/O

定义 4.5 (直接 I/O) *Linux* 的系统调用：*open*, *close*, *lseek*, *fsync*, *read*, *write*, *pread*, *pwrite* ……

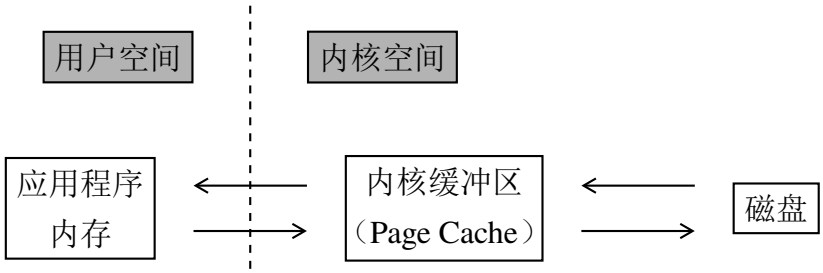

图 2: 直接 I/O

从缓冲 I/O 和直接 I/O 的示意图中可以看出：一次读写操作，缓冲 I/O 需要 3 次拷贝，直接 I/O 需要 2 次拷贝。

注

“直接 *I/O*”中“直接”的意思是没有用到用户缓冲区，但是内核缓冲区还是用到的。

fflush：用户缓冲区–> 内核缓冲区；*fsync*：内核缓冲区–> 磁盘。

TODO：*pread/pwrite* 在多线程读写同一个文件时很有用。

4.2 内存映射文件与零拷贝

4.2.1 内存映射文件

在 Linux 中对应的系统调用是 `mmap`。

应用程序的逻辑内存地址直接映射到 Linux 操作系统的内核缓冲区，实际读写的是内核缓冲区。

注

TODO：*try mmap*，测试进程实际内存占用量。

我之前的误区：误以为 *mmap* 是将应用程序的逻辑内存映射到物理内存。

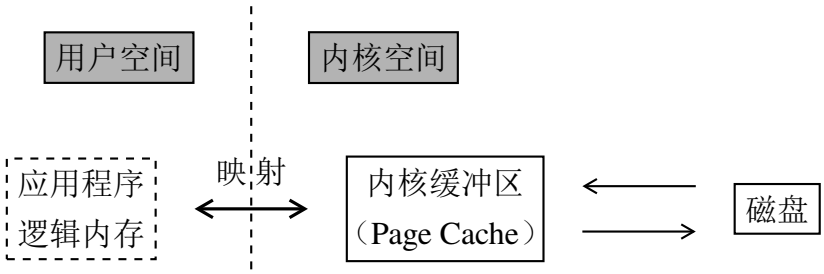

图 3: 内存映射文件原理

4.2.2 零拷贝

需求：把文件中的数据发送到网络。

约束：尽量减少数据在内存中拷贝的次数。

1. 实现方法 1：直接 I/O，整个过程需要 4 次数据拷贝。

```
fd1 = 文件描述符
fd2 = socket 描述符
buffer = 应用程序内存
read(fd1, buffer...) // 先把数据从文件读到应用程序内存
write(fd2, buffer...)   // 再通过网络发送出去
```

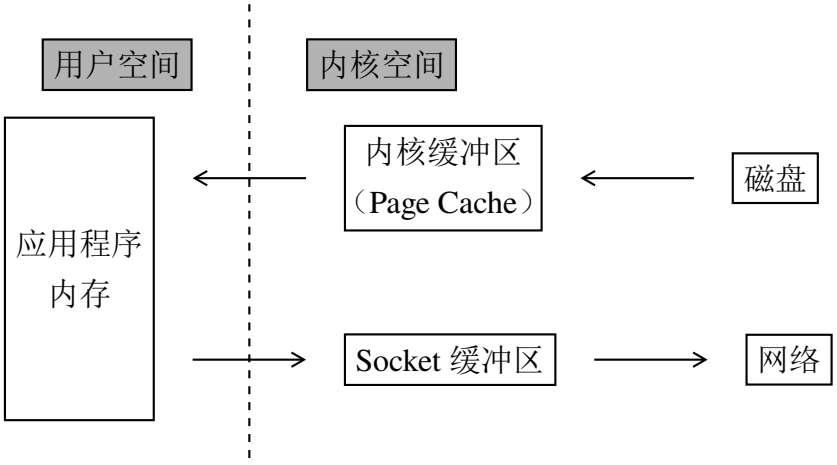

图 4: 直接 I/O 实现文件的网络发送

2. 实现方法 2：内存映射文件，整个过程需要 3 次数据拷贝。

```
fd1 = 文件描述符
fd2 = socket 描述符
buffer = 应用程序内存
mmap(fd1, buffer...) // 先把磁盘数据映射到 buffer 上
write(fd2, buffer...)   // 再通过网络发送出去
```

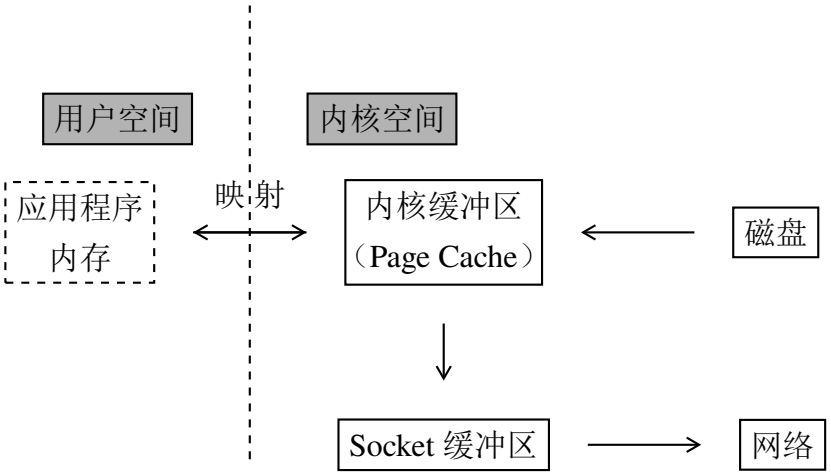

图 5: 内存映射文件实现文件的网络发送

3. 实现方法 3：零拷贝，整个过程需要 2 次数据拷贝。

```
fd1 = 文件描述符
fd2 = socket 描述符
sendfile(fd2, fd1...)   // 通过 Linux 系统调用实现零拷贝
```

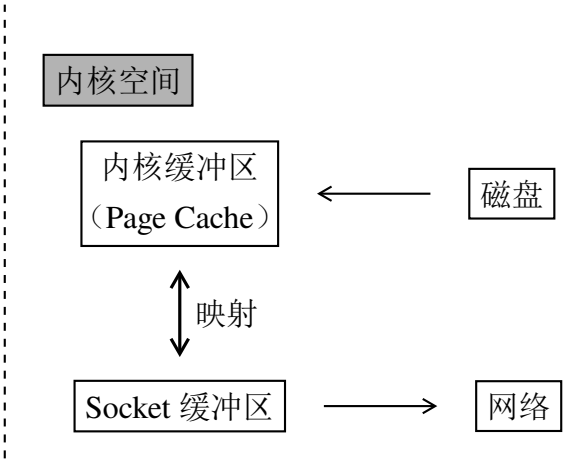

图 6: 零拷贝

注

“映射”和“拷贝”的区别：拷贝是将数据从一块内存复制到另一块内存；映射相当于持有数据的引用（或地址），数据本身只有一份。

“零拷贝”是指数据在内存中不存在拷贝，只存在内存和 *I/O* 之间传输。

Kafka 消费消息的时候利用了零拷贝技术。

4.3 网络 I/O 模型

区分操作系统层面和应用框架层面，底层和上层。

4.3.1 实现层面的网络 I/O 模型

概念纠正：

- 非阻塞 I/O（Non-Blocking I/O）≠ 异步 I/O（asynchronous I/O）
- I/O 多路复用（select、poll、epoll）≠ 异步 I/O（asynchronous I/O）
- 不存在“异步阻塞 I/O”这种说法

Linux 操作系统中的网络 I/O 模型：

表 2: 网络 I/O 模型

分类	I/O 模型	具体实现
同步 I/O	外部招式	内功心法
	行（实践）	知（理论）
异步 I/O	答案	问题

1. 同步阻塞 I/O
2. 同步非阻塞 I/O
3. I/O 多路复用
4. 异步 I/O

4.3.2 Reactor 模式与 Proactor 模式

4.3.3 select、epoll 的 LT 与 ET

4.3.4 服务器编程的 1+N+M 模型

4.4 进程、线程和协程

17 团队能力的提升

18 模板设计

此模板设计的初衷是为了记录笔记，在 2013 年开始构想，初版我们设计了非常美观的定理环境，并设计了 3 套不同的颜色主题。但我们发现在实际记笔记的时候，过多的定理区块使得整个文章并不是非常美观，所以我们把 ElegantNote 更新为 ElegantBook 模板，在后面被用户熟知。而 ElegantNote 的设计自此停止。

2018 年，在被一些用户“催更”之后，ElegantBook 迎来重大更新，原先浮动的定理环境用 tcolorbox 全部改写。时至今日，ElegantBook 版本为 3.05。之后，我们便想把 ElegantNote 也彻底更新下，放弃 ElegantBook 中的定理环境设计，改用更为紧凑，更加朴素的定理环境，设计更适合笔记记录和笔记阅读的 LaTeX 模板。

在一些朋友的建议和启发下，我们基于标准的 LaTeX 文类 article 重新设计了新版 ElegantNote 模板，在此特别感谢！新模板有下面几个特性：

- 添加护眼模式，颜色为绿豆沙颜色，默认为白色背景；
- 适配不同设备，包括 Pad（默认），Kindle，PC（双页），通用（A4）；
- 5 套颜色主题，分别是：green（默认），cyan，blue，sakura，black；
- 语言模式支持：中文（默认），英文；
- 支持 PDFLaTeX 和 XeLaTeX 编译；
- 更加美观的图表标题格式，列表环境，数学字体等。

18.1 护眼模式

本模板增加了护眼模式，默认为不开启，开启的方法如下：

```
\documentclass[geye]{elegantnote} % or
\documentclass[mode=geye]{elegantnote}
```

评论 此次更新只添加了护眼模式（绿豆沙色背景），如果您有希望增加其他颜色，可以在 *Github/ElegantNote* 创建 *issues* 或者提交 *pull request*。

18.2 设备选择

为了让笔记方便在不同设备上阅读，免去切边，缩放等操作，本模板适配不同的屏幕大小，分别为 Pad，Kindle，PC，A4。不同屏幕的选择为

```
\documentclass[device=pad]{elegantnote}
\documentclass[device=kindle]{elegantnote}
\documentclass[device=pc]{elegantnote}
\documentclass[device=normal]{elegantnote}
```

注 也可以采取直接赋值的方法选择屏幕，比如：

```
\documentclass[pad]{elegantnote}
\documentclass[kindle]{elegantnote}
\documentclass[pc]{elegantnote}
\documentclass[normal]{elegantnote}
```

注 如果想要得到普通的 *A4* 纸张大小的 *PDF*，需要选择 *device=normal*，而不是选择 *device=pc*，因为 *device=pc* 实际上设置的是电脑双页模式。

18.3 颜色主题

本模板内置 5 套颜色主题，分别是 green，cyan，blue，sakura，black。其中 green 为默认颜色主题，如果用户不需要彩色，可以选择 black 主题。颜色主题的启用方法和之前一样：

```
\documentclass[green]{elegantnote}
\documentclass[color=green]{elegantnote}
...
\documentclass[black]{elegantnote}
\documentclass[color=black]{elegantnote}
```

18.4 语言模式

本模板内含两套语言环境，改变语言环境会改变图表标题的引导词（图，表），文章结构词（比如目录，参考文献等），以及定理环境中的引导词（比如定理，引理等）。不同语言模式的启用如下：

```
\documentclass[cn]{elegantnote}
\documentclass[lang=cn]{elegantnote}
\documentclass[en]{elegantnote}
\documentclass[lang=en]{elegantnote}
```

注 不管选用中文环境还是英文环境均可输入中文。另外如果在笔记中使用了抄录环境（*lstlisting*），并且其中包括了中文，请务必使用 *XeLaTeX* 编译。本说明文档也只能通过 *XeLaTeX* 编译。

18.5 编译方式

本模板支持两种编译方式，PDFLaTeX 和 XeLaTeX，选用 PDFLaTeX 编译的话，如果用到了中文，则会调用 ctex 宏包，而如果选用 XeLaTeX 编译的话，则会调用 xeCJK 宏包。模板测试环境为 Win10 + TEX Live 2018，设定的字体为 Windows 中的宋体、楷体、黑体、微软雅黑等。如果你的电脑是 Mac/Linux 系统，而且采用 XeLaTeX 编译的话，请把 elegantnote.cls 中字体改为自己系统的字体。

18.6 定理类环境

此模板采用了 amsthm 中的定理样式，使用了 4 类定理样式，所包含的环境分别为

- **定理类**：theorem，lemma，proposition，corollary；
- **定义类**：definition，conjecture，example；
- **备注类**：remark，note，case；
- **证明类**：proof。

评论 在选用 *lang=cn* 时，定理类环境的引导词全部会改为中文。

19 写作示例

我们将通过三个步骤定义可测函数的积分。首先定义非负简单函数的积分。以下设 E 是 $\mathcal{R}^n$ 中的可测集。

定义 19.1 (可积性) 设 $f(x)=\sum_{i=1}^{k} a_i\chi_{A_i}(x)$ 是 E 上的非负简单函数，其中 $\{A_1,A_2,\ldots,A_k\}$ 是 E 上的一个可测分割，$a_1,a_2,\ldots,a_k$ 是非负实数。定义 f 在 E 上的积分为

$$\int_E f dx=\sum_{i=1}^{k} a_i m(A_i). \tag{1}$$

一般情况下 $0\le\int_E f dx\le\infty$。若 $\int_E f dx<\infty$，则称 f 在 E 上可积。

一个自然的问题是，Lebesgue 积分与我们所熟悉的 Riemann 积分有什么联系和区别？之后我们将详细讨论 Riemann 积分与 Lebesgue 积分的关系。这里只看一个简单的例子。设 $D(x)$ 是区间 $[0,1]$ 上的 Dirichlet 函数。即 $D(x)=\chi_{Q_0}(x)$，其中 Q_0 表示 $[0,1]$ 中的有理数的全体。根据非负简单函数积分的定义，$D(x)$ 在 $[0,1]$ 上的 Lebesgue 积分为

$$\int_0^1 D(x)dx=\int_0^1 \chi_{Q_0}(x)dx=m(Q_0)=0 \tag{2}$$

即 $D(x)$ 在 $[0,1]$ 上是 Lebesgue 可积的并且积分值为零。但 $D(x)$ 在 $[0,1]$ 上不是 Riemann 可积的。

表 3: 燃油效率与汽车价格

	(1)	(2)
燃油效率	-238.90***	-49.51
	(53.08)	(86.16)
汽车重量		1.75***
		(0.641)
常数项	11,253***	1,946
	(1,171)	(3,597)
观测数	74	74
R^2	0.220	0.293

括号内为标准误

*** p<0.01, ** p<0.05, * p<0.1

定理 19.1 (Fubini 定理) 若 $f(x,y)$ 是 $\mathcal{R}^p \times \mathcal{R}^q$ 上的非负可测函数，则对几乎处处的 $x \in \mathcal{R}^p$，$f(x,y)$ 作为 y 的函数是 $\mathcal{R}^q$ 上的非负可测函数，$g(x) = \int_{\mathcal{R}^q} f(x,y)dy$ 是 $\mathcal{R}^p$ 上的非负可测函数。并且

$$\int_{\mathcal{R}^p \times \mathcal{R}^q} f(x,y)dxdy = \int_{\mathcal{R}^p} \left(\int_{\mathcal{R}^q} f(x,y)dy \right) dx. \tag{3}$$

证明. Let z be some element of $xH \cap yH$. Then $z = xa$ for some $a \in H$, and $z = yb$ for some $b \in H$. If h is any element of H then $ah \in H$ and $a^{-1}h \in H$, since H is a subgroup of G. But $zh = x(ah)$ and $xh = z(a^{-1}h)$ for all $h \in H$. Therefore $zH \subset xH$ and $xH \subset zH$, and thus $xH = zH$. Similarly $yH = zH$, and thus $xH = yH$, as required.

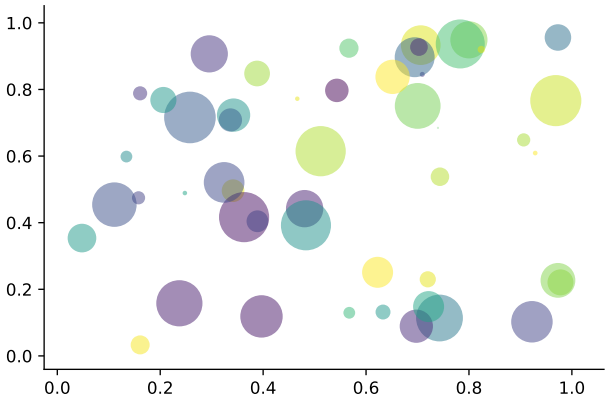

图 7: Matplotlib: Scatter Plot Example

回归分析（regression analysis) 是确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。根据定理 19.1，其运用十分广泛，回归分析按照涉及的变量的多少，

分为一元回归和多元回归分析；按照因变量的多少，可分为简单回归分析和多重回归分析；按照自变量和因变量之间的关系类型，可分为线性回归分析和非线性回归分析。

但是由于绝对值不易作解析运算，因此，进一步用残差平方和函数来度量总偏差。偏差的平方和最小可以保证每个偏差都不会很大。于是问题归结为确定拟合函数中的常数和使残差平方和函数最小。

- Routing and resource discovery;
  - Language Models
  - Vector Space Models
- Resilient and scalable computer networks;
- Distributed storage and search.

20 示例

```
\documentclass[geye,green,pad,cn]{elegantnote}

\title{Note Example}
\author{ddswhu}
\institute{Elegant\LaTeX{} Program}
\version{1.00}
\date{\today}

\begin{document}

\maketitle

\section{Introduction}
The content of Introduction.

\end{document}
```