05 | 思考题解读：如何实现可自我迭代的计算机？

2019-04-30 许式伟

讲述：姚迪迈
时长 13:16 大小 12.16M

你好，我是七牛云许式伟。

在第 3 讲“汇编：编程语言的诞生”中，我给出了一个架构思考题：

第一台以键盘 + 显示器为标准输入输出的现代计算机出现后，一个最小功能集，但计算能力可自我迭代的计算机应该是什么样的？

从需求上来说，我们期望它有如下能力。

键盘和显示器的驱动程序。

当时最主流的外置存储设备（不一定是现代的硬盘）的驱动程序。

一个汇编程序编辑器。可从存储中读取汇编程序代码，修改并保存到存储中。

一个汇编编译器。可将汇编程序代码编译成机器代码程序，并保存到存储中。

支持执行一段保存在外置存储设备中的机器代码程序。

那么，它的变化点和稳定点分别是什么？为此，你会怎么设计，设计出哪些子系统，每个子系统的规格是什么？扩展性上有哪些考虑？

需求分析

我们前面谈架构思维时提到：**做架构，第一件事情要学会做需求分析。**

需求分析的重要性怎么形容都不过分。准确的需求分析是做出良好架构设计的基础。我个人认为，架构师在整个架构的过程中，至少应该花费三分之一的精力在需求分析上。

这也是为什么很多非常优秀的架构师换到一个新领域后，一上来并不能保证一定能够设计出良好的架构，而是往往需要经过几次迭代才趋于稳定，原因就在于：领域的需求理解是需要一个过程的，对客户需求的理解不可能一蹴而就。

所以，一个优秀的架构师除了需要“在心里对需求反复推敲”的严谨态度外，对客户反馈的尊重之心也至关重要。只有心里装着客户，才能理解好需求，做好架构。

前面我们也强调过：在需求分析时，要区分需求的变化点和稳定点。稳定点往往是系统的核心能力，而变化点则需要对应地去考虑扩展性上的设计。

那么今天我们来实战一番，要实现一个最小化的计算能力可自我迭代的计算机，我们怎么做需求分析。

怎么实现可自我迭代的计算机?

通过前面对计算机工作原理的分析，我们已经知道，计算机分为三大类的零部件：

中央处理器；

存储；

输入输出设备。

中央处理器作为“计算”能力的核心，我们已经对它的工作范畴解剖清晰，这里不提。

存储，一方面作为“计算”的输入输出，另一方面作为“计算”本身的承载（也就是程序），主要的变数在后者。存储上的程序主要是：

计算机主板 ROM 上的启动程序（BIOS）；

外置存储上的软件。

接下来我们要考虑清楚的是：BIOS 负责做什么，外置存储上的软件负责做什么。这里我们先不展开。

输入输出设备，除了键盘和显示器外，还有外置存储。键盘和显示器我们只需要准备好对应的驱动程序，并没有特别需要考虑的内容。主要的变数在外置存储上。

外置存储在我们为它准备好了驱动程序后，就可以对它进行数据的读写了，但是我们接着需要考虑的问题是：我们准备把外置存储的数据格式设计成什么样？

回答这个问题前，先回顾下我们要做什么。目前我们已知的功能需求有如下这些。

键盘和显示器的驱动程序。

外置存储设备的驱动程序。

汇编程序编辑器。可从外置存储中读取汇编程序代码，修改并保存到外置存储中。

汇编编译器。可将汇编程序代码编译成机器代码程序，并保存到外置存储中。

支持执行一段保存在外置存储设备中的机器代码程序。

我们可以看到，外置存储需要保存的内容有：

汇编程序的源代码；

汇编编译器编译出来的可执行程序。

可见，外置存储它不应该只能保存一个文件，而是应该是多个。既然是多个，就需要组织这些文件。那么，怎么组织呢？

今天我们当然知道，操作系统的设计者们设计了文件系统这样的东西，来组织这些文件。虽然文件系统的种类有很多（比如：FAT32、NTFS、EXT3、EXT4 等等），但是它们有统一的抽象：文件系统是一颗树；节点要么是目录，要么是文件；文件必然是叶节点；根节点是目录，目录可以有子节点。

但是，文件系统（File System）是否是唯一的可能性？当然不是。键值存储（Key-Value 存储）也挺好，尤其是早期外置存储容量很可能极其有限的情况下。可以做这样统一的抽象：

每个文件都有一个名字（Key），通过名字（Key）可以唯一定位该文件，以进行文件内容的读写；

为了方便管理文件，可以对文件名做模糊查询（List），查询（List）操作支持通配符（比如我们现在习惯用的*和?）；

未来外置存储的空间有可能很大，需要考虑文件管理的延展性问题；可以考虑允许每个文件设定额外的元数据（Meta），例如创建时间、编辑时间、最后访问时间、以及其他用户自定义的元数据。通过元数据我们也可以检索（Search）到我们感兴趣的文件。

聊完了外置存储，让我们再回来看看 BIOS 和外置存储的软件怎么分工。

首先，BIOS 和外置存储上的软件分工的标准是什么？BIOS 是刻在计算机主板 ROM 上的启动程序，它的变更非常麻烦。所以 BIOS 负责的事情最好越少越好，只做最稳定不变的事情。

我们一一来看当前已知的需求。

首先是外部设备的驱动程序：键盘和显示器的驱动程序、外置存储设备的驱动程序。一方面，只要键盘、显示器、外置存储没有大的演进，驱动程序就不变，所以这块是稳定的；另一方面，它们是 BIOS 干其他业务的基础。所以，这个事情 BIOS 必然会做。

其次是汇编程序编辑器。编辑器的需求是模糊的，虽然我们知道它支持用户来编写程序，但是整个编辑器的操作范式是什么样的，没有规定。所以它不像是给键盘写一个驱动程序那样，是一个确定性的需求，而有很多额外的交互细节，需要去进一步明确。

你可以留意下自己日常使用的编辑器，去试着列一下它们的功能列表。你会发现小小的编辑器，功能远比你接触的大部分常规软件要多得多。

再次是汇编编译器。汇编编译器从输入输出来看，似乎需求相对确定。输入的是汇编源代码，输出的是可执行程序。但认真分析你会发现，它实际上也有很大的不确定性。

其一，CPU 会增加指令，这时候汇编指令也会相应地增加。对于大部分应用程序，CPU 新增的指令如果自己用不到，可以当它不存在。但是汇编语言及编译器需要完整呈现 CPU 的能力，因此需要及时跟进。

其二，虽然汇编指令基本上和机器指令一一对应，但是它毕竟是面向程序员的生产力工具，所以汇编语言还是会演进出一些高阶的语法，比如宏汇编指令。

所谓宏汇编指令，就是用一个命令去取代一小段汇编指令序列，它和 C 语言里面的宏非常类似。所以汇编语言并不是稳定的东西，它和其他高级语言类似，也会迭代变化。这就意味着汇编编译器也需要相应地迭代变化。

最后，执行一段保存在外置存储设备中的机器代码程序。这个需求看似比较明确，但是实际上需求也需要进一步细化。它究竟是基于外置存储的物理地址来执行程序，还是基于文件系统中的文件（文件内容逻辑上连续，但是物理上很可能不连续）来执行程序？

实现上，这两者有很大的不同。前者只需要依赖外置存储的驱动程序就可以完成，后者则还需要额外理解文件系统的格式才能做到。

那么，BIOS 到底怎么把执行控制权交到外置存储呢？

在学冯·诺依曼结构的时候，我们提到过，CPU 加电启动时，它会从存储的一个固定地址开始执行指令，这个固定地址指向的正是 BIOS 程序。

类似的，我们的 BIOS 也可以认定一个外置存储的固定地址来加载程序并执行，而无需关心磁盘的数据格式是什么样的。这个固定地址所在的数据区域，我们可以把它叫做引导区。

引导区的存在非常重要，它实际上是 BIOS 与操作系统的边界。

对于 BIOS 来说，执行外置存储上的程序能力肯定是需要具备的，否则它没有办法把执行权交给外置存储。但是这个能力可以是非常简约的。BIOS 只需要执行引导区的程序，这个程序并不长，完全可以直接读入到内存中，然后再执行。

我们是否需要基于文件系统中的文件来执行程序的能力？答案是需要。因为汇编编译器编译后的程序在外置存储中，需要有人能够去执行它。

综上，我们确认 BIOS 需要负责的事情是：

键盘和显示器的驱动程序；

外置存储设备的驱动程序；

支持执行外置存储中引导区的机器代码程序；

跳转到外置存储的固定地址，把执行权交给该地址上的引导程序。

而汇编程序编辑器、汇编编译器，以及支持执行文件系统中的程序，则不应该由 BIOS 来负责。

那么，外置存储上的引导程序拿到执行权后干什么呢？

我们再来总结下当前我们遇到的需求。

需要有人负责支持外置存储的数据格式，提供统一的功能给其他程序使用。无论它是文件系统，还是 Key-Value 存储系统。

需要有人提供管理外置存储的基础能力，比如查询（List）一下外置存储里面都有些什么文件。它可以实现为一个独立的程序，比如我们命名为 ls。

需要有人执行外置存储上的可执行程序。它可以实现为一个独立的程序，比如我们命名为 sh。

汇编程序编辑器。其实这个程序和汇编语言没什么关系，就是一个纯正的文本编辑器。我们可以把这个程序命名为 vi。

汇编编译器。它可以实现为一个独立的程序，比如我们命名为 asm。

引导程序拿到执行权后，我们不管它额外做了哪些事情，最终它要把执行权交给 sh 程序。因为，sh 程序算得上是可自我迭代的计算机扩展性的体现：通过 sh 程序来执行外置存储上的任意程序，这也相当于在扩展 CPU 的指令集。

结语

我们来回顾一下今天的内容。一个最小功能集、计算能力可自我迭代的计算机，它的变化点和稳定点分别是什么？为此，你会怎么设计，设计出哪些子系统，每个子系统的规格是什么？扩展性上有哪些考虑？

需求的变化点在于下面这几点。

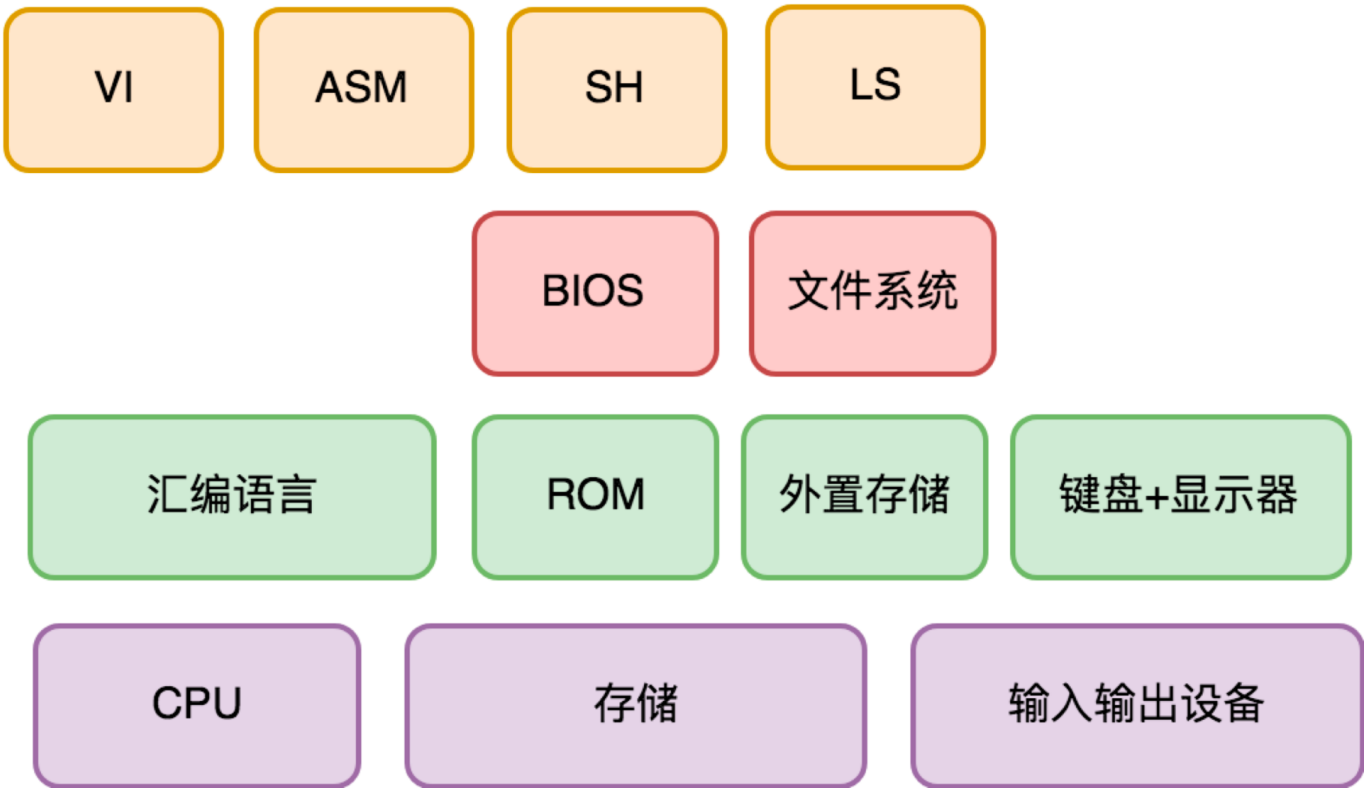
外置存储的数据格式。对此，我们设计文件系统（或 Key-Value 存储）子系统来负责这件事情。另外，我们也提供了 ls 程序来管理外置存储中的文件。

用户最终拿到这个计算机后，会迭代出什么能力。对此，我们设计了 sh 程序，让它支持在外置存储上执行任何应用程序。

编辑器的交互范式。对此，我们设计了 vi 程序，让它迭代编辑器的能力。

汇编语言的使用范式。对此，我们设计了 asm 程序，让它响应 CPU 指令集的迭代，以及汇编语言进化的迭代。

最终，我们设计出来的“可自我迭代的计算机”，它的系统架构看起来是这样的：



你的需求分析和系统设计跟上面的架构一致吗？

不一致非常正常，架构并无标准答案。但通过对比别人的方案与自己的不同之处，可以加深你对架构设计在决策上的体会。

另外，在 “可自我迭代的计算机” 这样相对模糊需求的细化过程中，也会很自然出现不太一样的理解，这些都是正常的，这也是需求分析的重要性所在，它本身就是一个需求从模糊到细化并最终清晰定义的过程。

如果你觉得系统过于复杂，不知道如何下手，也不要紧，设计“一个可自我迭代的计算机”的确是一个复杂的系统，它并不是一个非常适合架构新手的任务。但是我仍然希望通过这样一个例子的剖析，你对需求分析中稳定点和变化点的判断有所感悟。

如果你有什么样的想法和疑问，欢迎你给我留言，我们一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。感谢你的收听，我们下期再见。

上一篇 04 | 编程语言的进化

下一篇 06 | 操作系统进场

精选留言 (39)

写留言

Enthusiasm
2019-05-02

45

总结：设计系统架构的前提是用户需求分析，用户需求包括分析出稳定需求点和变化需求点。从功能上看，稳定需求点一般是实现偏核心需求的需求点，变化需求点往往是实现偏扩展性需求的需求点。从层次结构上看，稳定需求点往往在系统层次的底层，而变化需求点往往在更加抽象层（上层）。从从属关系上看，稳定点需要提供功能给变化点使用，变化点调用稳定点提供的功能。从时间顺序看，稳定需求往往先现是变化点实现的基础，...

展开

作者回复: 很赞的总结

梦里
2019-04-30

21

老师授课知识的角度很有深度，更贴切的说是一种思维方式，这种深度思考，从事情的本质重新推演与复盘的思考方式是很值得学习的。因为我们大部分人应该都没有想过自己去重新设计一个计算机的实现。

展开

作者回复: 从无到有到万物，我们这个课的脉络之一就是重新从零构建整个信息世界，这一点在开篇词中提到过，这一点非常非常关键。另一个脉络是架构思维的递进，这一章重点是需求分析。这两个脉络相辅相成，交织在一起。

s
2019-05-08

13

已经很久没有看到这么赞的文章了，起初踏入编程的大门就是想创造，在我眼里设计一个项目仿佛完成一个完美的艺术品。工作了以后感觉这份激情渐渐的被磨平，面对客户需求，仅仅是 "他指哪，我打哪"，软件的产出仅仅只是换钱的砝码。这篇文章又把我的梦想拉了回来。技术能解决的问题都不叫问题，唯独思想的升华是无价的，万分感激!

展开

作者回复: 这是我听到的最好的赞美，感谢。能够唤醒人们对架构之美的追求，无疑是我最大的动力。

WL
2019-04-30

10

请问一下老师, 03和05两篇文章中的"可自我迭代的计算机"中的"可自我迭代"是啥意思, 是指的是计算机可以执行各种各样的外部程序吗?

作者回复: 指功能可以越来越强大，是活的机器，而不是能力固定的机器

山口谈退隐...
2019-05-08

9

看完这篇文章让我想起了在一本育儿书上看到过这么一句话：“当人们使用‘需求’这个词的时候，他们所说的常常是一个能够满足需求的解决方案而不是需求本身”。而需求分析就是为了找到问题，然后告诉人们，你其实还可以这么做而不是你应该这么做。

作者回复: 赞，很多人都有这个误区，以解决方案代替需求。要清醒认知这一点并不容易。

Enthusiasm
2019-05-02

7

“活到老学到老”啊，突然意识到，不光是软件架构如此，连计算机系统架构也是如此。我们从小学计算机，每门计算机课程的第一课总是概述“计算机由硬件系统和软件系统构成”。
今天，许老师通过抽象出“稳定点”和“变化点”的概念，从另一个角度带给我对这句话更深的理解，才让明白这句话的真正含义。...
展开

作者回复: 硬件系统就是“稳定点”，软件系统就是“变化点”。挺好的总结。

裴海港
2019-04-30

6

我觉得拿一个软件系统的架构作为例子是不是更好，因为对于有些人来说，BIOS，ROM，文件系统，驱动程序这些过于抽象，难有代入感，不便于消化吸收。
展开

作者回复: 这个例子的确有点复杂。不过这些概念引入的时候都有相应的内容介绍，而且也是理解计算机很重要的基础，所以在例子类型上没有刻意避免。您可以说一下哪个概念比较难以理解，这样我可以多补充一下相关背景知识。

涵
2019-04-30

3

请问老师，最终架构图中的文件系统和现实中的操作系统是什么关系?文件系统是否是操作系统的一个子系统?或者说操作系统是否就是从文件系统中衍生出来的具有更多功能的文件系统?谢谢!

展开

作者回复: 挺好的问题。这里的文件系统子系统，它的交付物是什么？假设还没有操作系统这样的基础软件，它是否可以是包（package）？我觉得是可以的，虽然这样无法实现文件锁这样的多进程协同能力，但是我们实现的是单进程的计算机，不需要有文件锁这种东西。

Dimple
2019-04-30

3

拿到一个需求，需要做明确的需求分析，这是对我目前水平来说最有用的方式。文中讲的需求分析，让我受益很深。

不过实践部分，有点吃力，还需要细细品味才行。

作者回复: 做任何事情，首先要有正确的姿势，然后就是千锤百炼、熟能生巧了。

乘风
2019-05-11

1

老师写的确实是经典，自身功力不够不能了解其中的脉络。看了几遍，感觉懂了一些，又感觉没懂，对这几遍做一个总结吧：
1.需求分析是架构设计中最重要的事，只有彻底的分析清楚需求才能明白系统内部的部件和各部件之间的交互流程，清晰的展现系统工作流程。
2.如何去分析一个需求。首先考虑系统的整体，然后对整体进行细分，细分的过程还意味...

展开

猿工匠
2019-05-02

1

厉害👍，动手学习计算机结构设计和系统。学习了。

展开

大光
2019-05-01

1

渐入佳境，起初感觉讲的太缥缈，逐渐理解是为了从最原始，最根源，从无到有👍

ljf10000
2019-05-01

1

内置/外置存储是这样划分？内置存储应该是ram/rom/flash这些cpu可直接寻址的设备，其它不可直接寻址的设备是外置存储。CPU启动必然是从一个非易失的内置存储上某条指令开始。

展开

作者回复: 对的

Geek_88604...
2019-05-01

1

请问许老师，用户执行ls命令的时候，从系统架构图上看各模块之间是如何交互的？

作者回复: ls会调用文件系统，文件系统会调用外置存储的驱动程序，驱动程序会调用cpu的端口io指令。

明翼
2019-05-01

1

例子很复杂，和我设想的很不一样

展开

ljf10000
2019-04-30

1

1。这里外置存储应该包括我们平常所说的内存和硬盘。现在允许cpu直接执行存储上的指令，除了内存，可能大家熟悉的只有flash了。

2。如果没有cpu直接执行外置存储上指令的能力，就是有sh也没用，毕竟sh也是保存在外置存储上。

展开

作者回复: 1、不包括内存。外置存储（外存）是指cpu不直接支持的存储。cpu直接支持的叫内置存储，包括：寄存器、内存（RAM）、主板上的ROM。

2、cpu并没有直接执行外置存储上指令的能力。它通过支持虚拟内存做到支持外置存储的。虚拟内存在某一个页不存在的时候发生缺页中断，缺页中断由某个函数响应，再把缺的页从外存读进来。

gopherliu
2019-04-30

1

我突然想起个事，关于稳定点和变化点。各个银行的手机app大同小异，有必要每个银行都造一遍轮子？甚至就是所有的手机app都是基本的那一套。最为稳定的一点就是：账户密码。这玩意有必要每个app来一份吗？难道就不能有一个每个人单独的、唯一的数字身份？为了争夺用户，却给用户带来的更多的麻烦。还有，用户的数据为啥就不能属于用户自己的呢，被各大互联网巨头利用过来、利用过去的。对于此，许老师有何看法？

展开

作者回复: 账号密码，这个是个选择权的问题，确实是产品经理和架构师要考虑的点。现在蛮多软件都直接接微信或支付宝账号的。用户数据属于用户有点理想化，因为数据是在使用软件过程中产生的，更多只能通过法律途径解决，而挺难技术层面解决。

小智e
2019-04-30

1

老师的 level 确实很高，看问题的角度很到位，也很独特，学习了。

展开

糖果屋
2019-04-30

1

个人理解是对需求进行拆分，大的抽象需求拆分成多个明确的具体需求；再确定具体需求的变化点与稳定点。难点就是最初具体需求拆分粒度、需求接口规划、需求之间的管理等。随着系统的复杂度越来越高，需求的拆分也要随之变化。
学习老师的思想，期待老师的方法论

展开

jueyoq
2019-04-30

1

马斯克： 这就是我说的 第一性原理呀 许老师 么么哒

展开