50 | 技术分歧，如何决策？

2018-11-26 胡峰

作为一名程序员或技术人，总会碰到这样的场景：在一些技术评审会上，和其他程序员就技术方案产生分歧与争论。如果你是一名架构师或技术 Leader，站在技术决策者的立场和角度，该如何去解决分歧，做出决策呢？

这背后，有什么通用的方法和原则吗？

绝对

曾几何时，我以为技术是客观的，有绝对正确与否的标准判断。

在学校我刚开始学习编程技术时，捧着一本数据库教材，它在述说着经典的关系数据库表设计原则：第一、第二、第三范式。后来，我参加工作，那时的企业应用软件系统几乎都是以数据库为核心构建的，严格遵守范式定义的表结构。所以，当时觉得所有不符合范式设计的应用肯定都是错的，直到后来进入大规模的分布式领域，碰到了反范式设计。

也还是在学校做课程设计时，一起学习的同学总跟我讨论设计模式。一边写代码，一边研究这个代码到底符不符合某种模式，似乎没有套进某种模式中的代码就像没有拿到准生证的婴儿，带有某种天生的错误。直到后来，我碰到了反模式设计。

刚工作不久，同事和我讨论当用户删除自己的数据时，我们到底应不应该删掉它？我那时觉得理所应当写个 Delete 的 SQL 语句把它删掉。因为当时是这么想的：既然用户都不要他的数据了，我们还把它保留下来做什么呢？不是浪费资源嘛，而且服务器存储资源还算挺贵的。

但今天的互联网大数据时代，用户主动或非主动提交的任何数据，你都别想再将它真正地删除了。这个时代，受益于**摩尔定律**，存储设备容量不断增加，而价格不断降低，所有关于用户的数据总是可能有用的，都先存下来再说。

做技术这么些年下来，关于技术方案的判断，曾经以为的绝对标准，今天再看都是相对的。

相对

的确是的，适合的技术决策总是在相对的条件下做出的。

曾经，读到一篇英文文章，其标题翻译过来就是《简化：把代码移到数据库函数中》。我一看到这个标题就觉得这是一个错误的技术决策思路，为什么呢？因为曾经我花了好长时间做了一个项目，就是把埋在数据库存储过程中的代码迁移到 Java 应用里；而且，现在不依赖数据库的代码逻辑不正大行其道吗？

作者是在正话反说，还是在哗众取宠？我很是好奇。所以，我就把这篇文章仔细读了一遍，读完以后我发现作者说得似乎有些道理，他的说法我大概概括为如下。

作者说，如今绝大部分的 Web 应用包括两部分：

- 一个核心数据库，负责存储数据；
- 以及围绕数据库的负责所有业务智能与逻辑的代码，体现为具体编程语言的类或函数。

现在几乎所有的 Web 系统都是如此设计的，所以这像是真理，业界最佳实践，事实工业标准，对吧？但作者描述了他自己的经历，是下面这样的。

他从 1997 年开始做了一个电子商务网站，用了 PostgreSQL 作为数据库，第一版网站用 Perl 写的。1998 年换成了 PHP，2004 年又用 Rails 重写了一遍。但到2009年又换回了 PHP，2012 年把客户端逻辑拆出去用 JavaScript 重写，实现了前后端分离。

这么些年下来，代码重构过很多次，但数据库一直是 PostgreSQL。可是大量和数据存取有关的逻辑也随着代码语言的变迁而反复重写了很多遍。因而，作者感叹如果把这些与数据存取有关的逻辑放在数据库里，那么相关的代码将不复存在，他也不需要反复重写了。

这里有个疑问，作者没事老换语言，到底是在折腾啥？他虽然没有在文中明说，但作为程序员的我还是能设身处地感受到其中的缘由。作者本身是学音乐出身，目标是建网站卖音乐唱片，自学编程只是手段。作为一个过来人，我相信他早期的代码写得肯定不咋地，又在各种流行 Web 技术趋势的引诱下，充满好奇心地尝试各种当时时髦的技术，不断重构改进自己的代码。

在这个过程中发现，有一些和业务关系不太大的数据存取逻辑，被反复重写了很多遍，所以才产生出了这样的思路：假如把这部分代码移到数据库中。其实对这个思路的挑战，也是显而易见的：

- 如何进行调试、回滚？
- 如何做单元测试？
- 如何进行水平扩展？

上述“挑战”在一般情况下都成立，但对于作者来说却不是很重要。因为作者思路成立的前提是：第一，他维护的是一个小网站，数据库没有成为瓶颈；第二，这个网站的开发维护人员只有作者一个人，而不是一个团队。

是的，围绕这个网站，作者创办了一家公司，雇佣了 85 名员工，并成为了公司的 CEO 也是唯一的程序员。因此，这就是一个在作者所处特定环境下的技术决策，虽看上去明显不太对，但在作者的相对限定条件下，这个决策实际省了他个人的负担（虽然扩展有明显的极限，网站也不会发展太大）。

仔细看作者这个案例，可以发现其技术决策方案也是符合“康威定律”的。“康威定律”是这么说的：

> 任何组织在设计一套系统时，所交付的设计方案在结构上都与该组织的沟通结构保持一致。

换句话说，就是系统设计的通信结构和设计系统的团队组织的沟通结构是一致的。案例中，作者的系统只有他一个人负责设计与实现，只需要和不同阶段的自己产生沟通，在他的系统和场景下，变化最小、稳定度最高的是数据存储和结构，所以他选择把尽可能多的代码逻辑绑定在系统中更稳定的部分，从而降低变化带来的代价。

而**康威定律告诉我们系统架构的设计符合组织沟通结构时取得的收益最大**。这是一个经过时间检验和验证过的规律与方法，体现的就是一个相对的选择标准，那在这背后，有没有隐藏着关于技术决策更通用的判断原则呢？

原则

康威定律，是和组织的团队、分工、能力与定位有关的，其本质是**最大化团队的核心能力，最小化沟通成本**。

在足够大的组织中，沟通成本已经是一个足够大的成本，有时可能远超采用了某类不够优化的技术方案的成本。每一次人事组织架构变动的背后，都意味着需要相应的技术架构调整去适应和匹配这种变化，才能将沟通成本降下来。而**技术方案决策的核心，围绕的正是关于方案的实施成本与效率**。

曾经很多次的项目技术评审会上，后端的同学和前端的同学经常就一些技术方案产生争论。而争论的问题无所谓谁对谁错，因为同样的问题既可以后端解决，也可以前端解决，无论哪条路都可以走到目的地。那么还争论什么呢？无非是各自基于局部利益的出发点，让自己这方更省事罢了。

这些问题的解决方案处在技术分工的临界地带就容易产生这样的争论，而技术的临界区，有时就是一些无法用技术本身的优劣对错来做判断的区域。这时，最佳的选择只能是将前后端整体全盘考虑，以成本和效率为核心来度量，应该由哪方来负责这个临界区。

而**成本与效率背后的考量**又包括如下因素：

- **团队**：这是人的因素，关于团队的水平，掌握的技术能力和积累的经验；
- **环境**：能利用的环境支持，公司内部的平台服务或外部的开源软件与社区；
- **技术**：技术本身的因素，该项技术当前的成熟度，潜在的发展趋势；
- **约束**：其他非技术约束，比如管理权限的干涉、限定死的产品发布日期等。

不同的人，同样的技术方案，成本效率不同；不同的环境，同样的技术方案，成本效率也不同；不同的技术，同样的环境和人，成本效率也不同；不同的约束，同样的团队和环境，会得到不同的技术方案，成本效率自然不同。

在技术的理想世界中，技术决策的纯粹部分，其决策原则都和成本效率有关；而其他非纯粹的部分，其实都是“政治”决策，没有所谓通用的原则，只和博弈与利益有关。

最后，简单总结下：**技术没有绝对的标准，适合的技术决策，总是在受限的约束条件下，围绕成本与效率做出的选择权衡。对于一些纯粹的技术理想主义者，追求技术的完美与合理性，初心本不错，但也许现实需要更多的行动柔性**。

关于技术方案分歧，你是否遇到过类似的争论呢？又是采用的是怎样的决策方式？欢迎留言分享和大家一起讨论。
