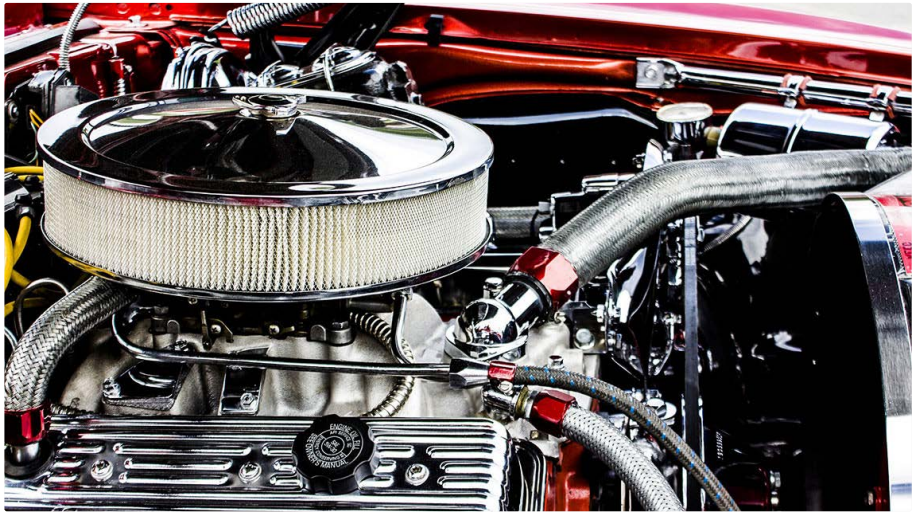
04 | 复杂度来源：高性能

2018-05-05 李运华



04 | 复杂度来源：高性能
李运华
00:00 / 16:17

周四，我为你讲了架构设计的主要目的是为了解决软件系统复杂度带来的问题。那么从今天开始，我将为你深入分析复杂度的6个来源，先来聊聊复杂度的来源之一高性能。

对性能孜孜不倦的追求是整个人类技术不断发展的根本驱动力。例如计算机，从电子管计算机到晶体管计算机再到集成电路计算机，运算性能从每秒几次提升到每秒几亿次。但伴随性能越来越高，相应的方法和系统复杂度也是越来越高。现代的计算机CPU集成了几亿颗晶体管，逻辑复杂度和制造复杂度相比最初的晶体管计算机，根本不可同日而语。

软件系统也存在同样的现象。最近几十年软件系统性能飞速发展，从最初的计算机只能进行简单的科学计算，到现在Google能够支撑每秒几万次的搜索。与此同时，软件系统规模也从单台计算机扩展到上万台计算机；从最初的单用户单工的字符界面Dos操作系统，到现在的多用户多工的Windows 10图形操作系统。

当然，技术发展带来了性能上的提升，不一定带来复杂度的提升。例如，硬件存储从纸带→磁带→磁盘→SSD，并没有显著带来系统复杂度的增加。因为新技术会逐步淘汰旧技术，这种情况下我们直接用新技术即可，不用担心系统复杂度会随之提升。只有那些并不是用来取代旧技术，而是开辟了一个全新领域的技术，才会给软件系统带来复杂度，因为软件系统在设计的时候就需要在这些技术之间进行判断选择或者组合。就像汽车的发明无法取代火车，飞机的出现也并不能完全取代火车，所以我们在出行的时候，需要考虑选择汽车、火车还是飞机，这个选择的过程就比较复杂了，要考虑价格、时间、速度、舒适度等各种因素。

软件系统中高性能带来的复杂度主要体现在两方面，一方面是单台计算机内部为了高性能带来的复杂度；另一方面是多台计算机集群为了高性能带来的复杂度。

单机复杂度

计算机内部复杂度最关键的地方就是操作系统。计算机性能的发展本质上是由硬件发展驱动的，尤其是CPU的性能发展。著名的“摩尔定律”表明了CPU的处理能力每隔18个月就翻一番；而将硬件性能充分发挥出来的关键就是操作系统，所以操作系统本身其实也是跟随硬件的发展而发展的，操作系统是软件系统的运行环境，操作系统的复杂度直接决定了软件系统的复杂度。

操作系统和性能最相关的就是进程和线程。最早的计算机其实是没有操作系统的，只有输入、计算和输出功能，用户输入一个指令，计算机完成操作，大部分时候计算机都在等待用户输入指令，这样的处理性能很显然是很低效的，因为人的输入速度是远远比不上计算机的运算速度的。

为了解决手工操作带来的低效，批处理操作系统应运而生。批处理简单来说就是先把要执行的指令预先写下来（写到纸带、磁带、磁盘等），形成一个指令清单，这个指令清单就是我们常说的“任务”，然后将任务交给计算机去执行，批处理操作系统负责读取“任务”中的指令清单并进行处理，计算机执行的过程中无须等待人工手工操作，这样性能就有了很大的提升。

批处理程序大大提升了处理性能，但有一个很明显的缺点：计算机一次只能执行一个任务，如果某个任务需要从I/O设备（例如磁带）读取大量的数据，在I/O操作的过程中，CPU其实是空闲的，而这个空闲时间本来是可以进行其他计算的。

为了进一步提升性能，人们发明了“进程”，用进程来对应一个任务，每个任务都有自己独立的内存空间，进程间互不相关，由操作系统来进行调度。此时的CPU还没有多核和多线程的概念，为了达到多进程并行运行的目的，采取了分时的方式，即把CPU的时间分成很多片段，每个片段只能执行某个进程中的指令。虽然从操作系统和CPU的角度来说还是串行处理的，但是由于CPU的处理速度很快，从用户的角度来看，感觉是多进程在并行处理。

多进程虽然要求每个任务都有独立的内存空间，进程间互不相关，但从用户的角度来看，两个任务之间能够在运行过程中就进行通信，会让任务设计变得更加灵活高效。否则如果两个任务运行过程中不能通信，只能是A任务将结果写到存储，B任务再从存储读取进行处理，不仅效率低，而且任务设计更加复杂。为了解决这个问题，进程间通信的各种方式被设计出来了，包括管道、消息队列、信号量、共享存储等。

多进程让多任务能够并行处理任务，但本身还有缺点，单个进程内部只能串行处理，而实际上很多进程内部的子任务并不要求是严格按照时间顺序来执行的，也需要并行处理。例如，一个餐馆管理进程，排位、点菜、买单、服务员调度等子任务必须能够并行处理，否则就会出现某个客人买单时间比较长（比如说信用卡刷不出来），其他客人都不能点菜的情况。为了解决这个问题，人们又发明了线程，线程是进程内部的子任务，但这些子任务都共享同一份进程数据。为了保证数据的正确性，又发明了互斥锁机制。有了多线程后，操作系统调度的最小单位就变成了线程，而进程变成了操作系统分配资源的最小单位。

多进程多线程虽然让多任务并行处理的性能大大提升，但本质上还是分时系统，并不能做到时间上真正的并行。解决这个问题的方式显而易见，就是让多个CPU能够同时执行计算任务，从而实现真正意义上的多任务并行。目前这样的解决方案有3种：SMP（Symmetric Multi-Processor，对称多处理器结构）、NUMA（Non-Uniform Memory Access，非一致存储访问结构）、MPP（Massive Parallel Processing，海量并行处理结构）。其中SMP是我们最常见的，目前流行的多核处理器就是SMP方案。

操作系统发展到现在，如果我们要完成一个高性能的软件系统，需要考虑如多进程、多线程、进程间通信、多线程并发等技术点，而且这些技术并不是最新的就是最好的，也不是非此即彼的选择。在做架构设计的时候，需要花费很大的精力来结合业务进行分析、判断、选择、组合，这个过程同样很复杂。举一个最简单的例子：Nginx可以用多进程也可以用多线程，JBoss采用的是多线程；Redis采用的是单进程，Memcache采用的是多线程，这些系统都实现了高性能，但内部实现差异却很大。

集群的复杂度

虽然计算机硬件的性能快速发展，但和业务的发展速度相比，还是小巫见大巫了，尤其是进入互联网时代后，业务的发展速度远远超过了硬件的发展速度。例如：

- 2016年“双11”支付宝每秒峰值达12万笔支付。
- 2017年春节微信红包收发红包每秒达到76万个。

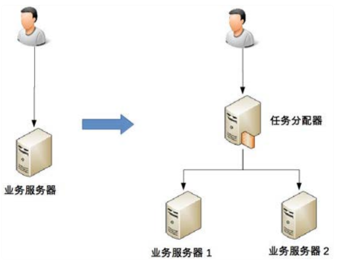
要支持支付和红包这种复杂的业务，单机的性能无论如何是无法支撑的，必须采用机器集群的方式来达到高性能。例如，支付宝和微信这种规模的业务系统，后台系统的机器数量都是万台级别的。

通过大量机器来提升性能，并不仅仅是增加机器这么简单，让多台机器配合起来达到高性能的目的，是一个复杂的任务，我针对常见的几种方式简单分析一下。

1.任务分配

任务分配的意思是指每台机器都可以处理完整的业务任务，不同的任务分配到不同的机器上执行。

我从最简单的一台服务器变两台服务器开始，来讲任务分配带来的复杂性，整体架构示意图如下。



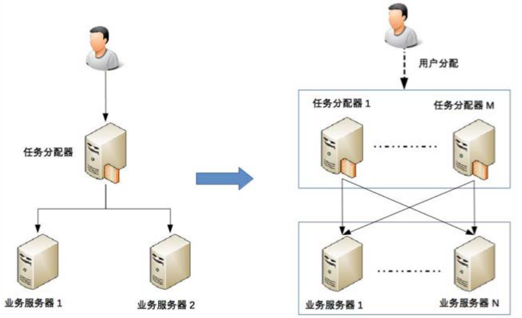
从图中可以看到，1台服务器演变为2台服务器后，架构上明显要复杂多了，主要体现在：

- 需要增加一个任务分配器，这个分配器可能是硬件网络设备（例如，F5、交换机等），可能是软件网络设备（例如，LVS），也可能是负载均衡软件（例如，Nginx、HAProxy），还可能是自己开发的系统。选择合适的任务分配器也是一件复杂的事情，需要综合考虑性能、成本、可维护性、可用性等各方面的因素。
- 任务分配器和真正的业务服务器之间有连接和交互（即图中任务分配器到业务服务器的连接线），需要选择合适的连接方式，并且对连接进行管理。例如，连接建立、连接检测、连接中断后如何处理等。
- 任务分配器需要增加分配算法。例如，是采用轮询算法，还是按权重分配，又或者按照负载进行分配。如果按照服务器的负载进行分配，则业务服务器还要能够上报自己的状态给任务分配器。

这一大段描述，即使你可能还看不懂，但也应该感受到其中的复杂度了，更何况还要真正去实践和实现。

上面这个架构只是最简单地增加1台业务机器，我们假设单台业务服务器每秒能够处理5000次业务请求，那么这个架构理论上能够支撑10000次请求，实际上的性能一般按照8折计算，大约是8000次左右。

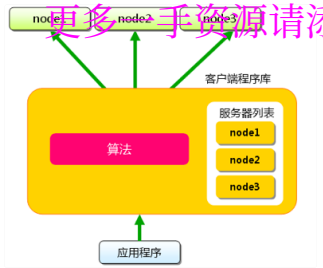
如果我们的性能要求继续提高，假设要求每秒提升到10万次，上面这个架构会出现什么问题呢？是不是将业务服务器增加到25台就可以了呢？显然不是，因为随着性能的增加，任务分配器本身又会成为性能瓶颈，当业务请求达到每秒10万次的时候，单台任务分配器也不够用了，任务分配器本身也需要扩展为多台机器，这时的架构又会演变成这个样子。



这个架构比2台业务服务器的架构要复杂，主要体现在：

- 任务分配器从1台变成了多台（对应图中的任务分配器1到任务分配器M），这个变化带来的复杂度就是需要将不同的用户分配到不同的任务分配器上（即图中的虚线“用户分配”部分），常见的方法包括DNS轮询、智能DNS、CDN（Content Delivery Network，内容分发网络）、GSLB设备（Global Server Load Balance，全局负载均衡）等。
- 任务分配器和业务服务器的连接从简单的“1对多”（1台任务分配器连接多台业务服务器）变成了“多对多”（多台任务分配器连接多台业务服务器）的网状结构。
- 机器数量从3台扩展到30台（一般任务分配器数量比业务服务器要少，这里我们假设业务服务器为25台，任务分配器为5台），状态管理、故障处理复杂度也大大增加。

上面这两个例子都是以业务处理为例，实际上“任务”涵盖的范围很广，可以指完整的业务处理，也可以单指某个具体的任务。例如，“存储”“运算”“缓存”等都可以作为一项任务，因此存储系统、运算系统、缓存系统都可以按照任务分配的方式来搭建架构。此外，“任务分配器”也并不一定只能是物理上存在的机器或者一个独立运行的程序，也可以是嵌入在其他程序中的算法，例如Memcache的集群架构。

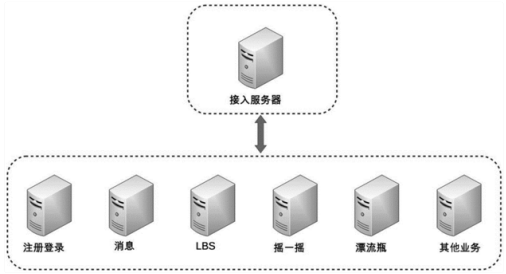


（http://my.oschina.net/uploads/space/2010/1015/163250_g2tS_98095.png）

2.任务分解

通过任务分配的方式，我们能够突破单台机器处理性能的瓶颈，通过增加更多的机器来满足业务的性能需求，但如果业务本身也越来越复杂，单纯只通过任务分配的方式来扩展性能，收益会越来越低。例如，业务简单的时候1台机器扩展到10台机器，性能能够提升8倍（需要扣除机器群带来的部分性能损耗，因此无法达到理论上的10倍那么高），但如果业务越来越复杂，1台机器扩展到10台，性能可能只能提升5倍。造成这种现象的主要原因是业务越来越复杂，单台机器处理的性能会越来越低。为了能够继续提升性能，我们需要采取第二种方式：**任务分解**。

继续以上面“任务分配”中的架构为例，“业务服务器”如果越来越复杂，我们可以将其拆分为更多的组成部分，我以微信的后台架构为例。



（http://image.jiagoushuo.com/2016/qAnayi.jpg）

通过上面的架构示意图可以看出，微信后台架构从逻辑上将各个子业务进行了拆分，包括：接入、注册登录、消息、LBS、摇一摇、漂流瓶、其他业务（聊天、视频、朋友圈等）。

通过这种任务分解的方式，能够把原来大一统但复杂的业务系统，拆分成小而简单但需要多个系统配合的业务系统。从业务的角度来看，任务分解既不会减少功能，也不会减少代码量（事实上代码量可能还会增加，因为从代码内部调用改为通过服务器之间的接口调用），那为何通过任务分解就能够提升性能呢？

主要有几方面的因素：

- 简单的系统更加容易做到高性能

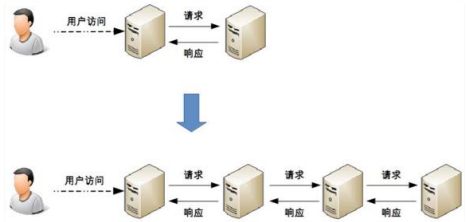
系统的功能越简单，影响性能的点就越少，就更加容易进行有针对性的优化。而系统很复杂的情况下，首先是比较难以找到关键性能点，因为需要考虑和验证的点太多；其次是即使花费很大力气找到了，修改起来也不容易，因为可能将A关键性能点提升了，但却无意中将B点的性能降低了，整个系统的性能不但没有提升，还有可能会下降。

- 可以针对单个任务进行扩展

当各个逻辑任务分解到独立的子系统后，整个系统的性能瓶颈更加容易发现，而且发现后只需要针对有瓶颈的子系统进行性能优化或者提升，不需要改动整个系统，风险会小很多。以微信的后台架构为例，如果用户数增长太快，注册登录子系统性能出现瓶颈的时候，只需要优化登录注册子系统的性能（可以是代码优化，也可以简单粗暴地加机器），消息逻辑、LBS逻辑等其他子系统完全不需要改动。

既然将一个大一统的系统分解为多个子系统能够提升性能，那是不是划分得越细越好呢？例如，上面的微信后台目前是7个逻辑子系统，如果我们把这7个逻辑子系统再细分，划分为100个逻辑子系统，性能是不是会更高呢？

其实不然，这样做性能不仅不会提升，反而还会下降，最主要的原因是如果系统拆分得太细，为了完成某个业务，系统间的调用次数会呈指数级别上升，而系统间的调用通道目前都是通过网络传输的方式，性能远比系统内的函数调用要低得多。我以一个简单的图示来说明。



从图中可以看到，当系统拆分2个子系统的时候，用户访问需要1次系统间的请求和1次响应；当系统拆分为4个子系统的时候，系统间的请求次数从1次增长到3次；假如继续拆分下去为100个子系统，为了完成某次用户访问，系统间的请求次数变成了99次。

为了描述简单，我抽象出来一个最简单的模型：假设这些系统采用IP网络连接，理想情况下一次请求和响应在网络上耗时为5ms，业务处理本身都为50ms；那么没有拆分系统的时候，对单个业务请求性能没有影响，那么系统拆分为2个子系统的时候，处理一次用户访问耗时为51ms；而系统拆分为100个子系统的时候，处理一次用户访问耗时就变成了

了149ms。

虽然系统拆分可能在某种程度上能提升业务处理性能，但提升性能也是有限的，不可能系统不拆分的时候业务处理耗时为50ms，系统拆分后业务处理耗时只要1ms，因为最终决定业务处理性能的还是业务逻辑本身，业务逻辑本身没有发生大的变化下，理论上的性能是有一个上限的，系统拆分能够让性能逼近这个极限，但无法突破这个极限。因此，任务分解带来的性能收益是有一个度的，并不是任务分解越细越好，而对于架构设计来说，如何把握这个粒度就非常关键了。

小结

今天我给你讲了软件系统中高性能带来的复杂度主要体现的两方面，一是单台计算机内部为了高性能带来的复杂度；二是是多台计算机集群为了高性能带来的复杂度，希望对你有所帮助。

这就是今天的全部内容，留一道思考题给你吧。你所在的业务体系中，高性能的系统采用的是哪种方式？目前是否有改进和提升的空间？

欢迎你把答案写到留言区，和我一起讨论。相信经过深度思考的回答，也会让你对知识的理解更加深刻。（编辑乱入：精彩的留言有机会获得丰厚福利哦！）

---

kingeasternsun

2018-05-05

大神能否在每篇文章结尾推荐一些相关的书籍，大神领进门，修行还是靠个人

公号-Java大后端

2018-05-05

今日心得
1 WHAT 对高性能的理解?
性能是软件的一个重要质量属性。衡量软件性能包括了响应时间、TPS、服务器资源利用率等客观指标，也可以是用户的主观感受（从程序员、业务用户、终端用户/客户不同的视角，可能会得出不同的结论）。

在说性能的时候，有一个概念与之紧密相关—伸缩性，这是两个有区别的概念。性能更多的是衡量软件系统处理一个请求或执行一个任务需要耗费的时间长短；而伸缩性则更加关注软件系统在不影响用户体验的前提下，能够随着请求数量或执行任务数量的增加（减少）而相应地拥有相适应的处理能力。

但是，什么是"高"性能？这可能是一个动态概念，与当前的技术发展状况与业务所处的阶段紧密相关。比如，现在在行业/企业内部认为的高性能，站在5年后来看，未必是高性能。因此，站在架构师、设计师的角度，高性能需要和业务所处的阶段来衡量。高到什么程度才能与当前或可预见的未来业务增长相匹配。一味去追求绝对意义上的高，没有太大的实际意义。因为，伴随性能越来越高，相应的方法和系统复杂度也是越来越高，而这可能会与当前团队的人力、技术、资源等不相匹配。但是什么才合适的高性能了？这可能需要从国、内外的同行业规模相当、比自己强的竞争者、终端用户使用反馈中获取答案并不断迭代发展。

软件系统中高性能带来的复杂度主要体现在两方面，一方面是单台计算机内部为了高性能带来的复杂度；另一方面是多台计算机集群为了高性能带来的复杂度。

2 WHY 为什么需要高性能？
追求良好的用户体验；
满足业务增长的需要。

3 HOW 如何做好高性能？
可以从垂直与水平两个维度来考虑。垂直维度主要是针对单台计算机，通过升级软、硬件能力实现性能提升；水平维度则主要针对集群系统，利用合理的任务分配与任务分解实现性能的提升。

垂直维度可包括以下措施：
增大内存减少I/O操作
更换为固态硬盘（SSD）提升I/O访问速度
使用RAID增加I/O吞吐能力
置换服务器获得更多的处理器或分配更多的虚拟核
升级网络接口或增加网络接口

水平维度可包括以下措施：
功能分解：基于功能将系统分解为更小的子系统
多实例副本：同一组件重复部署到多台不同的服务器
数据分割：在每台机器上都只部署一部分数据

垂直维度方案比较适合业务阶段早期和成本可接受的阶段，该方案是提升性能最简单直接的方式，但是受成本与硬件能力天花板的限制。

水平维度方案所带来的好处要在业务发展的后期才能体现出来。起初，该方案会花费更多的硬件成本，另外一方面对技术团队也提出了更高的要求；但是，没有垂直方案的天花板问题。一旦达到一定的业务阶段，水平维度是技术发展的必由之路。因此，作为技术部门，需要提前布局 ，未雨绸缪，不要被业务抛的太远。

三军

2018-05-05

面试官：小伙子，说下进程和线程？
我：
1， 早期的计算机是没有操作系统的，只有输入，计算，输出。手工输入速度远低于计算机的计算速度。

2， 于是出现了批处理操作系统，通过纸带，磁带等工具预先写入指令，形成一个指令清单（即任务）交给计算机处理。但批处理系统的缺点是只能有一个任务，而且当计算机在进行I/O处理时，CPU是空闲的。

3， 世人发明了进程，一个进程就代表一个任务，多个进程通过分时操作能让用户认为并行操作多任务，进程间的资源是独立单元，但是可以通过介质进行通信。缺点：进程内只进行串行处理，无法很好地分工合作提高处理效率。

4， 于是就有了操作系统调度的最小单元-线程，线程能够使进程内的子任务能够共享进程内的资源，并并行工作，大大提高操作系统的性能。

区别：
线程是任务调度的最小单元，共用进程内的资源。
进程是资源分配的最小单元，与其他进程资源互相独立。

探索无止境

2018-05-05

意犹未尽，期待后文！希望可以不受篇幅的限制，针对实战案例做更多的分析！

loveluckystar

2018-05-07

之前我们的系统是all-in的单系统模式，虽然水平扩展了大量机器，但是仍然存在性能问题，比如类似秒杀之类的活动，几乎会在瞬间把整个系统的数据库连接耗尽，导致其他服务发生卡顿甚至不可用；并且全在一个业务系统中，开发部属效率极低，扩展性也存在问题。

于是我们将系统进行了拆分，起初是按照业务拆分成几个核心系统，同时针对不同业务的负载情况进行了合理的水平扩展，整个系统的性能得到了提升，扩展性得到了保证，并且开发部署效率也得到了极大的提高。

但是随着业务的发展，之前的系统拆分不能满足现有业务，同时随着公司很多老员工的离开，之前的架构设计思路没有人清楚，于是就变成了走一步看一步的推进模式，衍生出了各种独立的服务达40个左右，这样系统之间的边界越来越模糊，甚至出现了服务间的循环调用，白白浪费时间。而且一次调用链路过长，发生问题很难定位。

所以我觉得我们的系统就是一个活生生的，没有搞好架构设计的例子，前期是没有设计导致性能瓶颈，后期是过度设计导致系统复杂。

作者回复

2018-05-07

典型案例，值得好好总结归纳一下

fiseasky

2018-05-06

李老师，当一个系统分为很多子系统时，每个子系统都有独立的数据库，如何保证数据的一致性呢？比如我有一个业务需要在A库插入一跳数据，在B库也要插入了一跳数据，然后在C库修改一条数据。假设中间那个库操作失败了，如何做到这个数据的一致性呢？

作者回复

2018-05-06

BASE原理，最终一致性，后面会讲

木木

2018-05-05

讲得很好啊，就是更新太慢，不够看啊

彡工鸟

2018-05-05

举例分析可否适当引入存储层来讲解呢。这才是真正的复杂点吧？

gevin

2018-05-21

我这边很多项目都是面向传统行业国企的，他们成熟的传统方案都和IT无关，先现在要向IT靠拢。通常用户那边的业务量、并发量小，企业不差钱，所以一般都是通过硬件层面的垂直扩展来提高性能的。对我们的用户而言，一方面喜欢性能强悍的硬件设备，另一方面，当我们给他们写软件开发的报告时，什么样的技术方案火，就要在报告里体现出什么样的技术（比如现在给用户的方案都要和向微服务靠拢），面子上的工作要做足，也很有意思~

作者回复

2018-05-21

这就是你们项目的复杂度：如何以更低成本优雅的装逼��

十七

2018-05-05

目前系统按业务做了拆分，确实带来了更大的复杂度，特别是数据库层面上，数据并不能根据业务完全分离

SadieS·?

2018-05-17

这个小程序可以改进一下吗，把语音的进度条提供出来。中断后不想从头听起

zeus2_18921421203

2018-05-05

目前性能首先必须把单机性能用起来，比如多线程一起执行，写入批量 减少io。单机到极限后用集群，集群必须要有任务调度，还存在互斥锁，复杂度急剧提高，性能再不够要分析性能瓶颈了，是io还是线程切换还是中断？基本单机加集群能搞定大部分，很少要优化线程模型的，用线程池就足够了，还有actor这个大杀器没用呢。

但莫

2018-05-05

我做的系统实现高性能的一种方式是缓存。把计算结果缓存起来，下次相同的请求直接从缓存获取结果就可以了。可以使用多级缓存，如本地缓存和集中缓存服务。
我在做系统的时候，应用层的扩展和优化相对持久化层的架构设计来说要容易一些。请问老师如何针对应用层和持久化层来做设计，有什么好的策略可以分享。
谢谢

老王

2018-05-05

我在做一个机器学习的程序，目前还是在单机上训练SVM，可以通过划分训练集，利用多台机器并行训练然后再合并的方式提升训练性能。

lm书生

2018-05-13

李老师，文章写得很棒，受益匪浅！但有一点疑惑的地方，通篇读完似乎没有看到具体的对高性能的定义，到底什么是高性能？高性能可以具体由哪些指标来衡量？是否遗漏了，还是说已经包含在行文里了，需要读者提炼？

作者回复

2018-05-14

高性能没有绝对标准，有的业务每秒处理1000就是高性能，有的每秒处理10000性能都一般，从业务出发，结合行业经验就可以判断性能要求是否高

印宏宇

2018-05-06

一、所有的高性能都要针对不同的业务场景：
（1）单机的高性能主要在于多任务处理，让不同或相同的任务能够同时处理。
（2）集群的高性能主要在于任务（业务模块，组件，资源）的拆分和利用，一是分配，二是分解，大规模分配的前提是要合理的分解系统，合理的分解也是为了更好的聚合任务。

二、其实现在针对网络访问，dns，反向代理，web服务，应用服务，缓存，数据库，分布式文件等都有很多的解决方案，但如何把这些方案和当前你的业务结合起来，并且你的业务如何进行设计，如何拆分功能和组件来满足不同时期的业务性能变化。

三、其实老师今天提供了一个渐进性的方案，也是业务性能变化的应对方法论，先单机，再集群，然后拆分再集群，不管什么样的性能问题都能用这个方法论来解决。具体如何集群相应开源的解决方案很多，但是业务系统如何拆分，拆分的粒度，我觉得老师也应该有相应的方法论，期待…

具体到我们公司，涉及到性能要改的地方很多了，包括静态资源处理，业务消息队列多线程处理，数据库现在是单库，如何进行读写分离…

Colin

2018-05-06

之前拆分过WMS系统，按业务划分，商品信息管理，收货模块，发货模块，仓储操作模块等等，项目更清晰，性能扩展方便，性能更好

作者回复

2018-05-06

谁用谁知道，一般人我们都不告诉他����

新人小胖

2018-05-05

我们现在的系统是一个消息处理系统，主要的瓶颈在于消息的处理是必须要是顺序的，不能乱序，所以subscribe消息是单线程的，目前需要解决这个问题。

作者回复

2018-05-06

“顺序”有两种场景：1. 按先后顺序分配，2. 处理完前一条才能处理完后一条。

第一种情况按照简单的任务分配就可以实现高性能，第二种如果任务的处理比较复杂的话，可以用任务分解，将任务分解为多个步骤，采用流水线的架构设计达到高性能，如果任务很简单，单台机器做好优化性能也能做到比较高，例如redis就是单进程的

清泉

2018-05-05

说说我的理解，不管是任务分配还是任务分解，都是通过分摊单台机器的流量来提高整个系统的处理性能。

对于任务分解，我认为不但没有性能上的收益，反而有性能上的损耗，本来可以在一个进程内部完成的交互，分解后却需要进行服务器间的网络交互。（分解前后业务逻辑不变的情况下）

不知道我这么理解对不对，但是与楼主说的任务分解一定程度上可以提高性能有些矛盾，求楼主指点迷津��

作者回复

2018-05-06

有性能损耗，但性能收益更多，举个简单例子，A功能和B功能在同一系统，A功能慢查询导致整个系统性能低，B功能性能同样被拉低。
我举例是告诉你说有慢查询，实际上很多系统隐藏的性能问题并不明显就能看出来。

李志博

2018-05-05

刚做了一个拆分是梳理核心接口和非核心接口，拆成两个项目，保证非核心接口出问题影响不到核心接口

pavel

2018-05-22

感谢老师回复。我们系统的量，每天学到存储大概有2T，采用mysql和hbase做存储。我们是做网站统计的，类似cnzz。每天接到的pv请求会到十亿次以上。我们使用集群接收，Kafka做消息队列，storm实时消费。统计结果存储在mysql，行为数据等存储在hbase。由于实时性要求以及量大，存储性能实在是一个瓶颈，主从同步滞后也相对严重，现在都已经去掉从库了。针对这种IO场景，而且实时性要求较高，R如何来应对呢？之前有个方案是mysql采取大量的分表分库，总共20台服务器，ssd硬盘，这样是能支持，但是成本还是挺大的，是否有更好的，或者我应该从哪方便去考虑呢？

作者回复

2018-05-22

1. 压缩
2. 合并：将多个数据合并为一个数据，可以在web端做
3. 采样：统计其实不需要精确值，例如1000000001和1000000002没有区别，可以用采样来推算原始值
你的系统复杂度就是大数据量（规模）和实时性，对结果其实不要求非常精确。

肖一林

2018-05-18

拆分业务就是消除木桶效应。服务之间调用尽量少，能减少系统损耗。

朱伟

2018-05-17

关于微信的架构，是实战经验总结还是自己推算？感觉不太对啊

作者回复

2018-05-17

微信自己公布的资料，你觉得哪里不对，我们可以探讨一下

J

2018-05-15

之前的公司用的是任务分解，现在的公司用的是任务分配，请教一下 ，文中的任务分解无需任务分配器么？比如ng?

作者回复

2018-05-15

可以用nginx做路由，也可以请求方直接访问具体的子系统

勇闯天涯

2018-05-08

工作中主要偏重前端开发，对老师讲的这些虽然理解(收获也很大)，但工作经验中并没有实践过，希望老师能推荐一些基础的实例的上手项目供我们这些初级菜鸟练习的机会。

作者回复

2018-05-08

先讲道和术，再给案例

探索无止境

2018-05-08

如何科学估算一个系统会承受的并发量，从而估算我们需要部署的服务器数量，这方面是否可以指点下？感谢！

huangshuihua

2018-05-06

果然大神。这个梳理和总结气到了醍醐灌顶的作用！

作者回复

2018-05-07

每篇都有人醍醐灌顶，我很开心����

itperson

2018-05-06

现在做的系统就是负载均衡加几台机器 文件服务器没有用到 缓存也没有用到什么 更别提集群了 我想了解如何评价系统负载能力 希望得到指教 感觉现在做的很多工作都没法评估实际负载能力

smiledou123

2018-05-06

谢谢老师，149ms问题已经明白，不过对于您在文章中提到的一些计算机原理知识我并不是太懂，能请您为像我一样的初级运维推荐一些需要看的计算机书籍吗？

作者回复

2018-05-06

我看的都是程序员的基础书籍吧，《unix网络编程卷1》、《tcp/ip协议卷1》、《unix高级编程》、《linux系统编程》、《高性能mysql》、《深入java虚拟机》

卡莫拉内瑟

2018-05-06

我们公司做的政府项目，没有高并发的场景，业务大多也是crud，高可用是有的，高扩展的场景较少，需求基本上是产品经理整理好的，一台ng，两台应用服务器，一主两从mysql，nas设备，redis都可以不用，请问这样业务场景的公司是否适合长期呆下去，还是说可以为了架构而架构，公司本身不差钱，给政府做项目几乎也是友情价，老板在乎的可能是数据

作者回复

2018-05-06

职业选择不是本专栏的内容呢，看你个人追求什么了，有的人追求稳定，有的人追求兴趣，有的人追求回报

闭嘴

2018-05-05

看完大神写的这篇文章，感觉主要是针对互联网的系统的架构，传统的企业级的架构思路难道也是要学习互联网的这种架构方式?

作者回复

2018-05-06

别急，后面会讲

曹铮

2018-05-05

任务分解提升性能的第一个因素有些牵强，除非是这一组任务对资源的依赖特性上差异很大时，才可以通过资源依赖的特性进行针对的优化，但这就属于第二个因素的范围了，否则，通过拆分能达到的效果，在一台机器上也能通过良好的编码达到

作者回复

2018-05-06

这个因素是普遍存在的，简单的系统确实更加容易做到高性能。

淡彩

2018-05-05

目前的业务系统主要是将用户界面、服务、数据库分层，除数据库外其他两层的服务都走LB用HAProxy。因为业务量不大，所以性能瓶颈暂时还不存在，不过服务层没有做到完全无状态化，有一些坑。本节内容主旨业务划分，节点扩张，后面应该慢慢就走向SOA，microservice了

作者回复

2018-05-05

本节的拆分是为了性能，不是为了业务可扩展，后面会讲可扩展

笑地

2018-05-05

目前做到了除数据层的其他各层各节点无状态且能任意扩展，下一步重点是将数据层按合适的粒度拆分多个块。我们系统的是现代ERP，关于数据层的拆分不知道大家有没有很好的办法或者合适的工具?

乘风

2018-07-17

1.模块分离，职责分离
2.根据模块的特性采用不同的机器，集群
3.加缓存，将延迟保证在可接受的范围
4.对于用户无感知的接口，采用mq，并行结果直接返回，后台串行或并行计算
5.主从+分表

Carlos

2018-07-11

我所在业务系统高性能采用的是任务分解加分布式集群的方式。我们所开发的系统基本是给运营商网络工程师使用的，系统最关键的复杂度在于高性能，虽然也做了安全性，高可用的架构设计，但是这不是系统主要瓶颈点所在，大多数时候仅仅是为了投标或者交付前期给客户讲讲系统设计的有多牛逼，后期很少用得到。
据我了解系统的用户主要分为两类，一类是运行商高层，CTO，CTO什么的，这些人会关注系统小时，天，周，月几个粒度的关键KPI报告，了解网络运行情况；另一类为网优人员，这些可能需要每天，或者每周，通过系统报表关联分析，定位分析已知网络问题，并给出优化解决方案。从用户特点可知，系统偶尔宕机，不会对这些客户工作造成大的影响，只要保证做好客户安抚，在尽可能短的时间恢复系统，并将丢失实际补齐即可，所以高可用计算不是系统架构设计的关键复杂度所在。安全性架构也不是关键复杂度所在，因为性能管理系统中包含的敏感数据不多，也不是黑客的主要攻击点，但对于特定几个国家，根据国家安全要求，安全性是首要要求。可扩展性架构我理解不论在哪个阶段都是应该考虑的，方便后期新功能加入和考虑系统容量变大时，无缝切换到其他架构，这一块我们当前系统仅在系统接入侧做了可扩展设计，但是系统内部就相当凌乱不堪了。最关键还是如何保障数据尽可能高的实时性。
我们的系统从大的方向上分为前端和后端。前端的功能主要是报表，Dashboard，告警等可视化模块，我们通常都把这些叫Web，被部署在web容器中，例如jboss，并在仅有的一台web服务器运行，这不是系统能关键性能瓶颈所在；后端功能主要包括数据采集解析和汇总计算，这两个模块是目前系统关键性能瓶颈点所在，并分别采用了分布式的架构设计。数据采集解析按照分布式设计分别被部署在多台服务器上，这些服务器被命名为EAM服务器，包含一台Master和多台Slave,Master机器主要负责任务生成和下发，不具备任务分配功能，任务该在哪台机器上运行，其实是认为划分好的，不用通过负载来自动分配，Slave负责执行实际的采集解析工作。数据汇总计算类似采集解析也被部署在多台服务器上，这些服务器被命名为APP服务器，包含一台Master和多台Slave,Master负责任务决策和任务分配，Slave负责任务执行。数据汇总计算子系统使用了Hadoop大数据生态圈技术，汇总计算本身使用Storm流式处理机制，不同任务被分配到不用worker中来执行，执行完数据存入Hbase，不同类型任务之间通过Kafka消息进行驱动。另外还有汇总计算过程中使用到的模型数据被缓存到Redis中，方便获取。
写到这里系统结构已经非常清晰，根据老师讲解，我来谈谈对这个架构的评价以及存在的问题。这个系统对复杂度判断正确，高性能架构方案也很优秀，但是即使这样在现场使用中还是被各种性能问题所困扰。我理解问题在于公司没有一套标准的编码规范，和代码审查机制，开发人员对编程语言特性，设计模式，内存管理，业务流程理解不够透彻，仅以完成功能开发为目的。导致一线客户在使用过程出现KPI计算不准确，为空等情况，开发人员定位后是由于代码不满足某些特性场景导致，然后又由另外不熟悉业务的开发人员修改代码，导致代码本身攀枝末节，错综复杂，这也是系统内部功能不具备可扩展性的原因。再写一个典型性能问题的发生场景，以某个汇总计算任务为例，第一步收到Kafka消息，第二步根据Kafka消息里的条件字段到oracle中查询Hbase文件key，第三步使用文件key到Hbase读取指定文件，文件中包含网元，时间，指标三部分。第四部，使用for循环读取文件每一样，每个单元格进行求和或平均。最后把汇总结果生成文件存入Hbase,并把Key存入Oracle。这样当客户现网网元数量增加，开启指标增加，for循环次数也跟着增加，性能问题就产生了。或者下层网管系统数据延迟，上个周期，或者上几个周期数据没来，在某一个时刻统一上来了，导致系统在某段时间负荷就会加大，接下来几个周期数据实时性都没发得到比较好的保障。

作者回复

2018-07-16

架构是系统质量的基础，但好的架构不能保证系统质量，和编码也有很大关系

---

yoummg

2018-07-08

总结一下老师所说的。
单机的高性能，更多的落脚点是机器本身的性能，以及处理具体任务的方式，多进程或多线程，更合理的应用单机的性能；
集群的高性能，即任务的分配和分解。分配是物理层面的，即哪些机器处理用户的请求；分解是业务层面的，即单个请求哪些业务子系统来处理。分配的难点：分配机器的性能，分配算法，与业务系统的连接，失败重试机制等等。分解难点在于业务切分的粒度把握，只能对业务与机器这两块理解恰到好处，才能做较好的切分。
因此集群的高性能的基础是把握好单机器性能。

hadooper

2018-07-05

任务分解：类似微信摇一摇等，后面还是会把摇一摇用任务分配的方式通过某种路由规则分配到某台业务服务器的，总觉得任务分解举例是不是hadoope 的map reduce更贴切一些?

作者回复

2018-07-05

map reduce更加难理解��

---

hadooper

2018-07-05

任务分解：类似微信摇一摇等，后面还是会把摇一摇用任务分配的方式通过某种路由规则分配到某台业务服务器的，总觉得任务分解举例是不是hadoope 的map reduce更贴切一些?

作者回复

2018-07-09

map-reduce是把大任务分解为多个相同的小任务，任务分解是将原来集成在一个系统中的多个不同任务分开为独立的任务

Carlos

2018-06-28

李老师，既然一个系统，实际架构设计中如何做到适当兼容比如数据库有Oracle，SQL Server2008，是怎么知道Oracle具备哪些特性，而SQL Server不具备，怎么做两个数据库性能对比测试等？

张国胜

2018-06-28

两种方式都在使用。先是使用了任务分配，在阿里云上采购一个负载均衡，负载均衡的后段是四台业务处理服务器，然后对业务做了任务分解，在每一台业务处理服务器上面都部署了多个分级后的任务模块。以达到满足当前性能的要求。当前我们的性能处理大概是500次每秒的请求。

作者回复

2018-06-28

如果单纯为了性能，四台服务器的规模，任务分解可能有点过度设计，如果是为了稳定性等那就没问题

水月洞天

2018-06-10

我们目前的系统，高性能提现在静态资源获取的性能和业务投标的高性能。前者我们才用了资源压缩节省空间的方式和分布式内容分发cdn。后者考虑到瞬时高并发问题，采用了内存缓存。由于当前用户在7000左右，还没有扩展为多服务器，接下来这是我们改进的方向。考虑到实现首先我们会考虑业务拆分。

作者回复

2018-06-10

我感觉不用为了性能扩展，为了可靠性扩展可以考虑一下

火山飘雪

2018-06-04

做传统银行系统的，现在做的系统感觉很不合理，一是所有的东西都堆在一个系统中，经常出现上传下载图片的小功能出站问题，导致整个系统很慢；二这个系统和其它系统见存在循环调用的情况，比A系统先调用B系统，B系统再反调A系统业务才算完成。

作者回复

2018-06-06

赶紧拆分，按照任务分解的方式，拆分成几个子系统

代码狂徒

2018-06-04

您好，针对文章中任务分配到任务分解的地方不太明白，任务分解的时候将下层业务系统进行了分解降低了复杂度，但上层任务分配器是如何降低复杂度的呢？流量依然是通过上层分流的吗？这样不是更复杂的网状结构吗？

作者回复

2018-06-04

任务分配器没有降低复杂度，而是为了高性能引入了复杂度

Tony

2018-05-28

醍醐灌顶，受益匪浅！
我反复咀嚼了3遍！

Joker

2018-05-24

Redis单进程还是做了取舍，毕竟支持的数据格式比mc复杂，如果多进程务必增加数据一致性的复杂度，实现起来就不那么容易了，但是单进程对于多核机器无法达到耗死的状态！

作者回复

2018-05-25

所以我们只能一台机器部署很多redis

pavel

2018-05-22

我们的业务系统，简单的按业务分为用户模块，后台管理模块，公众号模块。其中，用户模块细分为用户信息管理，用户账号管理，报表查看。之所以这么分，第一是为了职责分离，开发效率提升。第二是将报表等重IO查询的分离出来，以后好做扩展。目前存在的问题是数据层的分离。我们有大量数据需要实时写入和查询，这块如何优化呢？老师能否指导下。

作者回复

2018-05-22

首先，你要量化"大量"到底是多少，有的人认为一天1G就很大了，有的人认为1天1T才算大量
其次，实时写入和查询有很多方案，mysql，es，hbase都可以，关键是要符合你的场景，例如：写入后还能修改么？是什么样的查询？实时要求有多高？.....等等，需要你根据业务场景分析选择

孙振超

2018-05-20

除了前面讲了些性能的问题，后面更多的是关于系统容量问题，性能应该是说这一次请求的耗时，高性能意味着耗时的变短，而关于这一部分涉及的不多，通过添加更多的服务器、业务拆分等方式在提升系统容量的同时请求的耗时其实是变长了。

从雅京

2018-05-19

个人感觉任务拆分是一门艺术，很多公司都是为了拆分而拆分，边界划分不合理，随着项目推进模块之间开始交融或者重复调用

作者回复

2018-05-20

架构设计很多地方都是艺术��

交叉路口

2018-05-19

redis 单进程单线程？��只知道redis 操作原子性

作者回复

2018-05-20

redis的原子性就是因为是单进程

missa

2018-05-18

目前使用的任务拆分的方式，在每个拆分出来的任务中会有任务分配这种。

Yangjing

2018-05-16

拆分主要是为了维护 扩展，降低系统复杂度。这个拆分粒度应该怎么去考虑呢？ 有没有指标或者判断的标准？

作者回复

2018-05-16

后面会讲

志新

2018-05-15

总结：
高性能带来的是业务复杂性的提升

业务复杂提升，可将业务进行拆分达到简单的目的
拆分不可过细，带来的影响是各个系统之间调用过多导致性能下降

Sruby

2018-05-15

估值核算系统，性能瓶颈在估值核算部分。原来的是单体应用，现在把估值核算相关任务与中台应用拆分开了。估值核算中的很多任务调度使用xxl-job做分布式任务调度，充分利用现有的6台机器性能。

日光倾城

2018-05-12

你所在的业务体系中，高性能的系统采用的是哪种方式？目前是否有改进和提升的空间？
业务模块拆分，各模块按业务量大小决定单机/集群部署，各模块之间通过dubbo调度。目前也是感觉模块分得太细，调用复杂

summer

2018-05-11

提问：任务分解可带来高性能这点论证有些牵强。任务分解可以便于发现性能瓶颈,便于修改性能瓶颈。“便于发现”，“便于修改”都不是提高性能本身吧？

作者回复

2018-05-11

简单的系统更容易做到高性能

anchor

2018-05-09

一般的企业服务做到服务集群部署，数据库读写分离，已经能支撑一定业务量了，再后面分库分表，服务拆分都难有这业务量，而且做这些人力等资源不小

作者回复

2018-05-10

大部分情况做到这个已经够了

快乐的小傻子

2018-05-09

疑问1:代码结构相同，应用代码common相同的两个业务，是放在一起好呢？还是分开来处理好呢？

赵启发

2018-05-09

介绍线程的时候，只是提到因为单进程内部也需要并行，所以需要多线程，感觉还不够充分。比如文中餐饮管理进程的例子，排位、点餐需要并行，也可以引入多进程来实现。为什么需要线程，我觉得还是要提及多线程和多进程效率对比，比如线程/进程创建的开销，多线程/进程相互通信的开销等。

作者回复

2018-05-09

确实如此，但我的本意不是对比线程和进程，而是说明为什么有了进程后还要有线程。

关于多线程和多进城对比，后面架构模式部分会讨论

小飞哥 超級會員

2018-05-09

我是搞电商系统的，经历了从一个系统拆分出十个子系统。在老系统中一两个方法就分成一个系统，为了是业务相互不影响和对人员的深入，对业务理解和系统稳定而拆分。系统与系统的相互调用没发生改变，在每个系统的业务处理上得到扩展！，

DullBird

2018-05-08

1. 平台利用的是多台设备任务分解即集群的方案，
分为一个任务分解器（服务端）：仅仅实现调度和任务分配（轮询）
前期由平台抽象的任务，即各种采集项目封装为一个任务。
采集机集群，可以处理各种采集任务。

后期采集任务之间互相之间有逻辑和顺序越来越复杂，
然后任务分解器没有逻辑处理模块，导致现在只能尽量将任务封装成一个。
考虑需要在任务分解器集成逻辑模块。

Kevin

2018-05-08

老师可以讲一些典型电商的架构不，比如秒杀等等

作者回复

2018-05-09

这类案例互联网一搜一大把，我是希望通过我的架构方法论，能让大家真正理解这些案例，从而能够让自己也能设计出类似架构，而不是简单照搬其它架构

勇闯天涯

2018-05-08

总结一下所得，性能复杂度主要体现在单台计算机和多台计算机集群两方面。单机复杂度主要考虑多进程、多线程、进程间通信、多线程并发等技术点。集群主要考虑任务分配和任务分解，并且要把握好粒度，逼近性能极限。

ZYCHD(子玉)

2018-05-08

我们的系统是公司内部的业务系统，系统出现性能瓶颈大多只是在某个时间段出现，一年也就出现过几次。需要您说的这种架构吗？您介绍的架构很好，不过假如我们系统也这样做就显得资源浪费了。

作者回复

2018-05-08

首先看是否满足业务需求，不要过度设计

孙晓明

2018-05-08

一、体会：软件系统为了高性能可能会带来复杂度的增加，需在高性能与复杂度直接进行权衡，综合考虑成本、运维、风险等问题，避免为了追求高性能导致系统复杂度太高。
二、我接触过的系统中，在高性能方面：1、数据库采用集群设计，也有的时候使用读写分离；2、缓存服务器单独部署；3、应用服务器分层开发和部署，并使用集群；4、负载均衡采用过F5、Weblogic等。

sonald

2018-05-08

100个子系统耗时就149ms吗？取决于互联方式和调用模式吧。如果一个子系统并发的从其他99个子系统请求，就少于149了吧。您这里单指子系统串联的情况吗？

海德曼

2018-05-08

感谢老师回复。目前似乎系统不支持直接回复之前的问题解答，所以只能重新提问了。听您讲解之后感觉一些宏观的原理是明白了，但是如果要自己短时间搭建一个高性能的系统似乎还有不小的距离，下一步该怎么学习？能短时间突破吗？

海德曼

2018-05-07

请问老师，对于一个电商网站，单台服务器的并发用户数能达到一万吗？十万？单台服务器的并发用户数，连接数计算的方法或者依据是什么？

作者回复

2018-05-07

不能简单参考一个数字，和业务复杂度有关，而且并发用户数和并发请求数不完全等同，基本上上万是很难的，因为mc和redis这种功能简单的中间件，并发请求数也就每秒3万左右

王磊

2018-05-07

我们的系统采用了一个基于netty的NIO框架VertX来用少量的线程服务更多的请求，提高线程利用率；使用Redis作为缓存服务器和预加载的方式，提高响应时间，减少数据库的访问；使用分布式锁控制相同请求只有一个落库查询；优化数据库，提高数据库查询性能。

作者回复

2018-05-07

VertX是异步编程框架吧？我觉得有点杀鸡用牛刀的感觉��

唐昂星

2018-05-07

举一个最简单的例子：Nginx 可以用多进程也可以用多线程，JBoss 采用的是多线程；Redis 采用的是单进程，Memcache 采用的是多线程
我的意思是说，在这个语言环境下，强调redis采用的是单线程的模式是不是好一些

作者回复

2018-05-07

嗯，没问题，能理解就可以了

唐昂星

2018-05-07

redis是单线程吧

作者回复

2018-05-07

单进程

曲大伟

2018-05-07

总体方法论不错，期待能深入到实战，对提高性能的相关方法和工具做个介绍。

作者回复

2018-05-07

后面有详细的章节介绍

darrykinger.com

2018-05-06

全球三个数据中心，都是单机，数据要保持一致，后期根据用户的请求扩展服务器，区域不同定位到不同数据中心(就近原则:智能DNS，CDn)等，会考虑使用好可用，前期有没有比较好的方案，让三个数据库一致保持一致？并能够进行监控处理？

带刺的温柔

2018-05-06

分析出系统的复杂度可以通过调度、拆分来提高性能，在一定的约束框架内比如说内存 CPU io 网络等通过各种方法比如说算法 数据结构等降低系统的复杂度简化系统来达到提高性能。不明白的是高性能有没有标准

作者回复

2018-05-06

满足业务需求就是高性能第一标准

碧海蓝天

2018-05-06

我们业务是新项目，现在是单机单块，但系统内部仿微服务做的模块化设计，与微服务的区别是模块之间是进程内接口调用而不是IPC。现在也做到了前后端分离，服务层无状态。后面的性能提升思路是单机做到极致，然后拆成微服务，再多实例。

聂明星

2018-05-06

本人架构初学者，对负载均衡单点问题和性能瓶颈问题想了解一下具体要怎么处理，对于单点问题我能想到的就是利用keepalived来达到负载均衡器高可用的目的，对于负载均衡性能瓶颈只能想到用DNS来分流，不知道我的方案是否可行，由于公司对于架构方面不是很重视只能自己在网上搜集资料，在这想了解一下大公司针对这类问题是怎么处理的是否有一套标准的处理方案和流程。

作者回复

2018-05-06

最后的几篇就是讲互联网大公司的标准技术栈

wq

2018-05-06

作者本文讲的是架构复杂度的原因之一，性能。文中讲了性能问题的来源和两个解决办法，1，任务分配。2，任务分解。

1，任务分配主要是集群，通过多台主机处理业务。这里的任务分配可以理解为，所有的业务在一台主机，把这样的主机复制多台。2，任务分解，任务分解指的是业务拆分，业务拆分可以更好的找到系统瓶颈，可以更有利于代码优化。

以上，就是我学到的，谢谢作者，谢谢。

小叶子

2018-05-06

我不是开发者，问题比较初级，感谢支持！
1.为了实现高性能需要进行任务分解（不同应用服务器，文件服务器，数据库）
2.为了实现各个服务器的独立弹性扩展，可用性，应用服务器不能保存数据，这个我理解不太好。

用户通过反向代理把请求输入，应用服务器一般都做哪些工作，帮忙举几个例子

smiledou123

2018-05-05

假设这些系统采用 IP 网络连接，理想情况下一次请求和响应在网络上耗费为 1ms，业务处理本身耗时为 50ms。我们也假设系统拆分对单个业务请求性能没有影响，那么系统拆分为 2 个子系统的时候，处理一次用户访问耗时为 51ms；而系统拆分为 100 个子系统的时候，处理一次用户访问耗时竟然达到了 149ms。这是怎么计算的？为何是149ms？我看不懂

作者回复

2018-05-06

100个子系统中间有99条网络连接

小破烂儿

2018-05-05

老师，我们公司目前给各家医院的某个科室做系统，每家医院除了公共需求外都存在一些个性化的需求，目前我们是开了很多分支，每家医院针对一个版本分支进行开发。新合作一家医院确定需求后找一个相似的版本分支再开出一个版本分支进行开发，能从架构设计上弄成一个产品级的系统吗，通过不同配置给各家医院进行使用，如果这样做，预期前体现到多长之类时间吧

作者回复

2018-05-06

后面可用性性能会涉及

Dylan

2018-05-05

高性能分为两方面：快-单任务处理速度要快；多-系统单位时间内处理的任务数量要多。快可以通过优化程序，提高硬件，更改操作系统参数等方式来提升；多除了上面的方面还可以通过增加机器，将任务分解到不同机器上处理来提高。任务如何分解，系统如何交互以及复杂度的主要方面都是需要架构师来做分析与权衡的。

1024

2018-05-05

单个机器为了高性能，所以出现了进程、线程，那现在出现的协程是如何提高性能的呢?

作者回复

2018-05-06

协程可以简单理解为用户态的线程，协程切换是用户态的任务切换，性能消耗比线程切换更低

松涛

2018-05-05

假如有以下业务场景：
X系统为企业系统，根据业务拆分出以下两个系统。
A系统为人事系统
B系统为绩效系统（考勤，项目，积分等等）

B系统的人事数据以及考勤等部分数据都是来自于拆分后的A系统，这些时候是不是不太合理呢？拆出来后系统的耦合度还是挺好的，但是根据业务情况，好像也没有错。不知华哥你怎么看？

作者回复

2018-05-06

考勤系统依赖人事系统是正确的，但别直连数据库，用接口进行访问

郭峰

2018-05-05

如何挖掘单机性能极限？如何挖掘分布式系统性能极限？这个会讲到吗？或者老师能推荐若干性能测试的好书吗？:D

作者回复

2018-05-05

具体某个领域技术细节不会涉及，这里面每个领域都足够写一本书了。
性能测试我做过，但并不专业，还请你看看能否找到专业人员请教一下。

Seven _ dong

2018-05-05

按业务拆分，也就是微服务吧，从方法调用变为服务间的调用，怎么处理错误等情况也是复杂度提升的一个表现

作者回复

2018-05-06

都是拆分，但拆分的维度和粒度不同

Seven _ dong

2018-05-05

按业务拆分，也就是微服务吧，从方法调用变为服务间的调用，怎么处理错误等情况也是复杂度提升的一个表现

zt_soft

2018-05-05

大神，传统软件开发的我主要面对是业务的复杂度，不知道这算架构师的工作么，还是那叫业务架构师��

作者回复

2018-05-05

肯定算，业务复杂度也不简单，后面会讲到

武坤

2018-05-05

计算机一次只能执行一个任务，如果某个任务需要从 I/O 设备（例如磁带）读取大量的数据，在 I/O 操作的过程中，CPU 其实是空闲的?

李艳超_Harry

2018-05-05

拆分粒度是一个问题。拆分时候有些功能放到a可以，放b、c、d也可以。这些拆分还涉及到团队和人员安排

张玮(大圣)

2018-05-05

1. 读了所有的评论，其实随着这几年微服务架构思想的就行，很多时候一些架构设计同学都会选择一步到位把系统按照业务做拆分，当然这里有粒度的把控，过粗过细都会导致问题，这里可以理解为任务分配+任务分解思想的运用

2. 性能优化点思路总结起来还是两步走:先单机后集群，如多线程、协程等编程模型的使用，缓存的使用，最终都是为了更快完成一次请求服务

3. 运华兄，点赞，加油，希望突破千万订阅者!

作者回复

2018-05-05

全中国程序员加起来都没这么多��

hello

2018-05-05

目前我们主要业务系统是主要通过堆机器的方式来提升服务处理性能，现在想通过进一步拆分服务的方式来解决一些问题。文中只是定性分析了拆分的准则，能不能分享些可实操量化的拆分经验？文中所提的拆分，是不是就是微服务化?

作者回复

2018-05-06

后面大量章节会讲高性能高可用怎么做

feifei

2018-05-05

以nginx做负载中心，集群化业务服务

张志攀

2018-05-05

曾经接触过两类系统，一类是读多写少的，这类系统高性能通过系统服务化拆份和多级分布式缓存来抗住高QPS另外一类是交易型统，我们做法是服务化拆份和分库分表来抗住高并发写的，数据一致性采用最终一致性。谢谢！！

wuxu

学习了，目前接触的大多是ngnix下挂接口服务和后台服务、mysql做个主从、再加个redis，基本就满足了当前业务。没有接触过根据业务结点作分层，不知道如何提高这方面的能力

成功

2018-05-05

这课讲的很深入而且落地，在这个互联网+，大数据的技术背景下，集群模式的多机复杂度越来越主流。我感觉两个点尤为重要，1)任务分配器(流控，HAproxy，nginx)的选型和设计(多对多)。2)就是任务的分解，如何最优的平衡业务级应用服务批和网络通信的关系，期待下一讲

万里晴空

2018-05-05

我们系统对性能要求很高，但是从以前的架构和业务设计，导致从程序上无法提升，现在处理方式就是多节点和增加物理机提升。只能临时解决，目前因为这个需要重构系统，听你讲解可以将系统内拆分出适当的几个子系统。业务是根据时间段来放开学生和老师填不同维度的数据，在某个阶段，每天的活跃量在10w以上（后期会很多），而这些数据有可能需求变更或新增维度，目前新架构设计出来的是用业务模型和布局来灵活可配置。从实现上研发前期可能会投入更多的时间是人工配置了，更别说权限和菜单的控制了。想换一种方式来解决这种问题，希望可以给一个灵感

---

escray

2018-05-05

首先把系统跑起来，然后尽可能的做单机性能优化，在单机成为瓶颈时，考虑集群的任务分配和任务分解。

一般人可能并没有机会去接触类似于淘宝或者微信这样量级的架构，从个人而言，我更关注单机性能优化。

除了传统意义上的单机优化，如果把应用部署阿里云这样的云平台上，单机性能优化是否有不同？

Mike Wang

2018-05-05

任务分解和分配都有用到，任务分配用到dns，nginx，haproxy，还有注册发现类的一些组件；任务分解按交易流程划分为多个服务使用grpc进行调用，主要有账户服务、交易服务，订单服务、推送服务等

蓝皮

2018-05-05

任务拆分并非越细越好，但是要怎么分析出当前系统拆多细是最好的呢？

更多一手资源请添加QQ/微信1182316662