数据中心作为一台计算机的设计

仓库规模机器的设计简介

计算机体系结构综合讲座

编辑

马克·希尔，威斯康星大学麦迪逊分校

《计算机体系结构综合讲座》出版了50到150页的出版物，涉及设计、分析、选择和互连硬件组件以创建满足功能、性能和成本目标的计算机的科学和艺术相关主题。

数据中心作为一台计算机的设计：仓库规模机器简介

Luiz André Barroso和Urs Hölzle
2009

功耗效率的计算机体系结构技术

Stefanos Kaxiras和Margaret Martonosi
2008

芯片多处理器架构：提高吞吐量和延迟的技术

Kunle Olukotun, Lance Hammond, James Laudon
2007

事务性内存

James R. Larus, Ravi Rajwar
2007

量子计算机对计算机架构师的影响

Tzvetan S. Metodi, Frederic T. Chong
2006

数据中心作为一台计算机：仓库级机器设计导论
作者：Luiz André Barroso和Urs Hölzle
网址：www.morganclaypool.com

ISBN：9781598295566 平装书

ISBN：9781598295573 电子书

DOI: 10.2200/S00193ED1V01Y200905CAC006

Morgan & Claypool Publishers系列出版物

计算机体系结构综合讲座

讲座 #6

系列编辑：Mark D. Hill，威斯康星大学麦迪逊分校

系列ISSN

ISSN 1935-3235 打印

ISSN 1935-3243 电子的

数据中心作为一台计算机的设计

仓库规模机器的设计简介

Luiz André Barroso和Urs Hölzle
Google公司

计算机体系结构综合讲座＃*6*

MORGAN & CLAYPOOL PUBLISHERS

摘要

随着计算越来越多地转移到云端，感兴趣的计算平台不再像一个披萨盒或冰箱，而是一个装满计算机的仓库。 这些新的大型数据中心与早期传统的托管设施非常不同，不能简单地视为一组共同放置的服务器。 这些设施中的大部分硬件和软件资源必须协同工作，以高效地提供良好的互联网服务性能，这只能通过对其设计和部署的整体方法来实现。 换句话说，我们必须将数据中心本身视为一个巨大的仓库级计算机（WSC）。 我们描述了WSC的架构，影响其设计、运行和成本结构的主要因素，以及其软件基础的特点。 我们希望它对今天的WSC的架构师和程序员以及未来可能在单个板上实现类似今天WSC的未来多核平台的人员有用。

关键词

计算机组织与设计，互联网服务，能源效率，容错计算，集群计算，数据中心，分布式系统，云计算。

致谢

虽然我们从过去几年参与谷歌基础设施设计和运营中获得了经验，但我们在这里报告的大部分内容是我们谷歌同事的辛勤工作、洞察力和创造力的结果。我们在这里涵盖的主题与我们的平台工程、硬件运营、设施、站点可靠性和软件基础设施团队的工作最相关，因此，我们对他们的经验表示特别感谢。 尽管他们只看到了相对不成熟的早期版本的文本，但Ricardo Bianchini、Fred Chong和Mark Hill提供了非常有用的反馈。 我们的谷歌同事Jeff Dean和Jimmy Clidaras也对早期草稿提供了广泛而特别有用的反馈。 感谢谷歌的Kristin Weissman和Morgan & Claypool的Michael Morgan的工作，我们能够免费电子化地提供这个讲座，这是我们接受这个任务的条件。 我们很幸运，Gerry Kane自愿提供他的技术写作才能，显著提高了文本的质量。 我们还要感谢Catherine Warner在不同阶段对文本的校对和改进。

我们非常感谢Mark Hill和Michael Morgan邀请我们参与这个项目，他们不知疲倦的鼓励和必要的催促，以及他们似乎无尽的耐心。

本书的这个版本得益于Christian Belady、Jeremy Dion、Robert Hundt、Artur Klauser、David Konerding、Mike Marty、Vijay Rao、Jordi Torres、Amund Tveit、Juan Vargas、Pedro Reviriego Vasallo和Kristin Weissman提供的反馈和修正。我们真诚地感谢他们的帮助。

读者注意

我们非常感谢您对我们手稿的任何想法、建议或修正。 我们计划经常修订这本书，并确保明确感谢任何可以帮助我们提高其实用性和准确性的意见。 提前感谢您花时间做出贡献。 请在http://tinyurl.com/wsc-comments提交您的反馈。

(http://spreadsheets.google.com/viewform?formkey=cmFrdHUxcTJ2SndCV2E4RzdTOTA0QXc6MA).

目录

第一章

引言

ARPANET即将迎来四十岁的生日，而万维网也接近二十周年纪念日。 然而，这两个里程碑所激发的互联网技术至今仍在不断改变行业和我们的文化，且没有放缓的迹象。 最近，随着诸如基于Web的电子邮件、搜索和社交网络等流行互联网服务的出现，以及全球范围内高速连接的增加，向服务器端或“云”计算的趋势加速发展。

越来越多的计算和存储正在从类似个人电脑的客户端转移到大型互联网服务中。 虽然早期的互联网服务主要是信息性的，但如今许多Web应用程序提供以前存在于客户端的服务，包括电子邮件、照片和视频存储以及办公应用程序。 向服务器端计算的转变主要是由于对用户体验改进的需求，例如易于管理（无需配置或备份）和无处不在的访问（只需一个浏览器），以及它为供应商提供的优势。 作为一项服务的软件可以加快应用程序的开发，因为对于软件供应商来说，进行更改和改进更简单。 供应商无需更新数百万个客户端（具有各种奇特的硬件和软件配置），只需在其数据中心内协调改进和修复，并且可以将其硬件部署限制在少数经过良好测试的配置上。 此外，数据中心的经济性使得许多应用服务可以以较低的每用户成本运行。 例如，服务器可以在成千上万个活跃用户（以及更多非活跃用户）之间共享，从而实现更好的利用率。 同样，计算本身在共享服务中可能变得更便宜（例如，多个用户接收的电子邮件附件只需存储一次而不是多次）。 最后，与桌面或笔记本电脑相比，数据中心中的服务器和存储更容易管理，因为它们受到单一、熟悉的实体的控制。

一些工作负载需要如此多的计算能力，以至于它们更适合于大规模计算基础设施而不是客户端计算。 搜索服务（Web、图片等）是这类工作负载的一个典型例子，但是像语言翻译这样的应用程序也可以更有效地在大型共享计算环境中运行，因为它们依赖于大规模语言模型。

向服务器端计算的趋势和互联网服务的爆炸性流行创造了一类新的计算系统，我们将其命名为仓库规模计算机，或*WSC*。这个名字旨在引起对这些机器最显著特征的关注：它们的软件基础设施、数据存储库和硬件平台的大规模。 这种观点与隐含假设一个程序在单台机器上运行的计算问题的观点不同。 在仓库规模计算中，程序是一个互联网服务，可能由数十个或更多个单独的程序组成，这些程序相互交互以实现复杂的终端用户服务，如电子邮件、搜索或地图。 这些程序可能由不同的工程团队实施和维护，甚至跨越组织、地理和公司边界（例如，混搭应用）。

为了运行这样大规模的服务所需的计算平台与披萨盒服务器或者冰箱大小的高端多处理器几乎没有任何相似之处。 这样的平台硬件由数千个独立的计算节点组成，配备相应的网络和存储子系统、电力分配和调节设备以及广泛的冷却系统。 这些系统的外壳实际上是一个建筑结构，往往与大型仓库无异。

1.1 仓库级计算机

如果规模是这些系统唯一的区别特征，我们可能会简单地称之为数据中心。 数据中心是由于共同的环境要求和物理安全需求以及维护的便利性而将多个服务器和通信设备放置在一起的建筑物。 从这个意义上说，仓库级计算机可以被视为一种数据中心。 然而，传统的数据中心通常托管大量相对较小或中等规模的应用程序，每个应用程序在一个独立的硬件基础设施上运行，并与同一设施中的其他系统解耦和保护。 这些数据中心托管多个组织单位甚至不同公司的硬件和软件。 在这样的数据中心中，不同的计算系统在硬件、软件或维护基础设施方面往往没有共同之处，并且往往彼此之间不进行通信。

WSC目前为谷歌、亚马逊、雅虎和微软的在线服务部门提供动力。 它们与传统数据中心有很大的不同：它们属于一个组织，使用相对均匀的硬件和系统软件平台，并共享一个常见的系统管理层。 与传统数据中心中主要运行第三方软件相比，大部分应用程序、中间件和系统软件都是内部构建的。 最重要的是，WSC运行较少数量的非常大的应用程序（或互联网服务），并且共享的资源管理基础设施允许进行重要的

部署灵活性。 同质性、单一组织控制和增强的成本效益要求推动设计师采取新的方法来构建和运营这些系统。

互联网服务必须实现高可用性，通常目标是至少99.99%的正常运行时间（每年约有一小时的停机时间）。 在大量的硬件和系统软件集合上实现无故障运行是困难的，而且由于涉及的服务器数量众多，这一任务变得更加困难。
虽然在一组10,000台服务器中理论上可能防止硬件故障，但这肯定会非常昂贵。 因此，WSC工作负载必须被设计成能够优雅地容忍大量组件故障，对服务水平的性能和可用性几乎没有影响。

1.2 强调成本效益

构建和运营大型计算平台是昂贵的，服务质量可能取决于可用的总体处理和存储能力，进一步推高成本并要求关注成本效益。 例如，在诸如Web搜索的信息检索系统中，计算需求的增长由三个主要因素驱动。

- 增加的服务受欢迎程度会转化为更高的请求负载。
- 问题的规模不断增长——每天都有数百万个网页增加，这增加了构建和提供Web索引的成本。
- 即使吞吐量和数据存储库可以保持不变，这个市场的竞争性质也不断推动创新，以改善检索结果的质量和索引更新的频率。 虽然一些质量改进可以通过更智能的算法单独实现，但大多数实质性的改进都需要为每个请求提供额外的计算资源。 例如，在一个还考虑查询中搜索词的同义词的搜索系统中，检索结果的成本要高得多——要么搜索需要检索与更复杂查询（包括同义词）匹配的文档，要么需要为每个词的同义词在索引数据结构中进行复制。

对于WSC设计中的成本效益，必须广泛定义以考虑成本的所有重要组成部分，包括托管设施的资本和运营费用（包括电力供应和能源成本）、硬件、软件、管理人员和维修费用。

1.3 不仅仅是服务器的集合

我们的核心观点是，支持当今成功的互联网服务的数据中心不再仅仅是一个杂乱无章的机器集合，它们是在设施中共同放置并连接在一起的。 在这些系统上运行的软件，如Gmail或Web搜索服务，执行的规模远远超过单个机器或单个机架：它们运行在由数百到数千个个体服务器组成的集群上。 因此，这个大型集群或服务器聚合体本身就是一台计算机，需要将其视为一个单一的计算单元。

设计WSC的技术挑战对于计算机系统架构师的专业知识同其他任何类型的机器一样重要。 首先，它们是一类由新的和快速发展的工作负载驱动的大规模机器。 它们的规模使得它们难以高效地进行实验或模拟；因此，系统设计师必须开发新的技术来指导设计决策。 故障行为、功耗和能源考虑在WSC的设计中具有更重要的影响，也许比其他较小规模的计算平台更重要。 最后，WSC比由单个服务器或小组服务器组成的系统更复杂；WSC对程序员的生产力提出了一个重大的新挑战，这个挑战可能比多核系统的编程更大。 这种额外的复杂性间接地源于应用领域的规模更大，并表现为更深层次和不均匀的存储层次结构（本章后面讨论）、更高的故障率（第7章）和可能更高的性能变异性（第2章）

本书的目标是向读者介绍这个新的设计领域，描述一些WSC的要求和特点，突出这个领域独特的一些重要挑战，并分享我们在Google内部设计、编程和运营WSC的经验。我们有幸既是WSC的设计者，也是平台的用户和程序员，这为我们提供了一个不同寻常的机会来评估产品的整个生命周期中的设计决策。 我们希望能够成功地传达我们对这个领域的热情，将其作为一个令人兴奋的新目标，值得广大研究和技术界的关注。

1.4 一个数据中心VS.多个数据中心

在本书中，我们将计算机定义为一个以数据中心为架构的设备，尽管互联网服务可能涉及到相距很远的多个数据中心。 有时候，多个数据中心被用作完全复制的同一服务，复制主要用于减少用户延迟和提高服务吞吐量（一个典型的例子是Web搜索服务）。 在这些情况下，一个给定的用户查询往往在一个数据中心内完全处理，我们的机器定义似乎是合适的。

然而，在用户查询涉及跨多个数据中心的计算的情况下，我们单一数据中心的重点并不是一个明显的选择。典型的例子是处理非易失性用户数据更新的服务，因此需要多个副本以实现灾难容忍性。 对于这种计算，一组数据中心可能是更合适的系统。 但我们选择将多数据中心的情景视为更类似于计算机网络。 这部分是为了限制本讲座的范围，但主要是因为数据中心内部和数据中心之间的连接质量巨大差距导致程序员将这些系统视为独立的计算资源。 随着这类应用程序的软件开发环境的发展，或者如果连接差距在未来显著缩小，我们可能需要调整我们的机器边界选择。

1.5 为什么WSC对你很重要

到目前为止，WSC可能被认为是一个小众领域，因为它们的规模和成本使得除了少数大型互联网公司之外，其他人都无法负担得起。 毫不奇怪，我们不认为这是真实的。 我们相信今天的大型互联网服务面临的问题很快将对更多的人群产生意义，因为许多组织很快就能以更低的成本购买类似规模的计算机。 即使在今天，低端服务器级别计算平台的有吸引力的经济性也使得数百个节点的集群可以在相对广泛的企业和研究机构范围内实现。 当与单个芯片上大量处理器核心的趋势相结合时，一个机架上的服务器很快就会拥有与今天的数据中心相当数量甚至更多的硬件线程。 例如，一个机架上有40台服务器，每台服务器都有四个8核双线程CPU，将包含超过两千个硬件线程。 这样的系统很可能在几年内对非常多的组织来说是可以负担得起的，同时还具有今天WSC的规模、架构组织和故障行为的特点。因此，我们相信我们构建这些独特系统的经验将有助于理解设计问题和编程挑战，以应对这些潜在的普及化的下一代机器。

1.6 WSC的架构概述

WSC的硬件实现在不同的安装中会有显著差异。

即使在像Google这样的单个组织内，不同年份部署的系统使用不同的基本元素，反映了行业提供的硬件改进。 然而，这些系统的架构组织在过去几年中相对稳定。 因此，以高层次描述这种通用架构是有用的，因为它为后续讨论提供了背景。

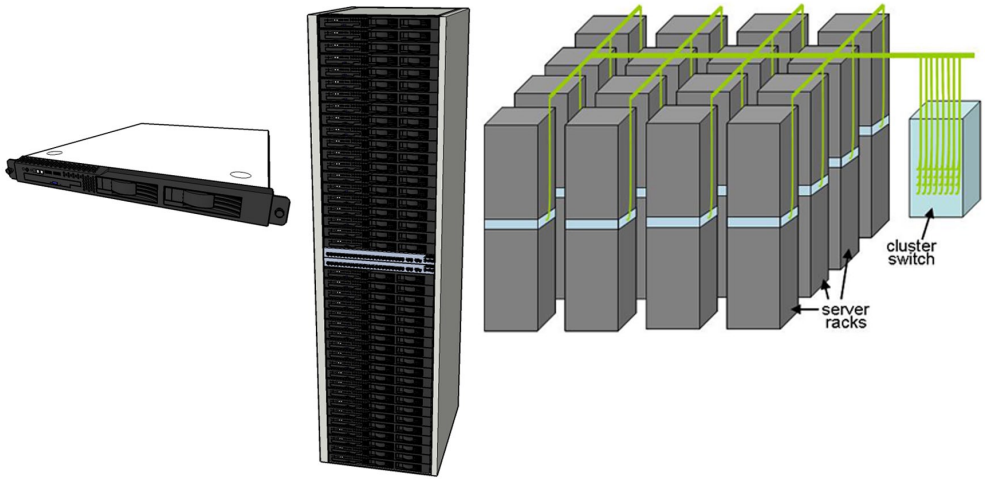

图1.1：仓库规模系统中的典型元素：1U服务器（左），带以太网交换机的7′机架（中），以及带有集群级以太网交换机/路由器的小型集群的图表（右）。

图1.1展示了一些更受欢迎的WSC构建模块。一组低端服务器通常以1U或刀片式机箱的形式安装在机架内，并使用本地以太网交换机进行互连。 这些机架级交换机可以使用1 Gbps或10Gbps的链路，与一个或多个集群级（或数据中心级）以太网交换机有多个上行连接。 这个第二级交换域可能跨越一万多个单独的服务器。

1.6.1 存储

磁盘驱动器直接连接到每个单独的服务器，并由全局分布式文件系统（如Google的GFS [31]）管理，或者它们可以是直接连接到集群级交换结构的网络附加存储（NAS）设备。 NAS倾向于是一个更简单的部署解决方案，因为它将数据管理和完整性的责任推给了NAS设备供应商。 相比之下，使用直接连接到服务器节点的磁盘集合需要在集群级别上使用容错文件系统。 这很难实现，但可以降低硬件成本（磁盘利用现有的服务器机箱）和网络结构利用率（每个服务器网络端口在计算任务和文件系统之间实际上是动态共享的）。 这两种方法之间的复制模型也有根本的不同。 NAS通过每个设备内的复制或错误校正功能提供额外的可靠性，而像GFS这样的系统则在不同的机器之间实现复制，因此

将使用更多的网络带宽来完成写操作。 然而，类似GFS的系统在整个服务器机柜或机架丢失后仍能保持数据可用，并且可以从多个副本中获取相同的数据，从而可能提供更高的聚合读取带宽。
为了降低成本、提高可用性和增加读取带宽，牺牲了更高的写入开销是谷歌许多工作负载的正确解决方案。 将磁盘与计算服务器放置在一起的另一个优势是它使分布式系统软件能够利用数据的局部性。
在本书的其余部分，我们将默认假设一个模型，其中分布式磁盘直接连接到所有服务器。

包括谷歌在内的一些WSC使用桌面级磁盘驱动器而不是企业级磁盘，因为两者之间存在显著的成本差异。由于这些数据几乎总是以某种分布式方式复制（如GFS），这可以减轻桌面级磁盘可能更高的故障率。 此外，由于磁盘驱动器的现场可靠性往往与制造商的规格有很大偏差，企业级磁盘的可靠性优势并不明确。

例如，Elerath和Shah [24]指出，与制造过程和设计相比，有几个因素可以更大程度地影响磁盘的可靠性。

1.6.2 网络结构

选择用于WSC的网络结构涉及速度、规模和成本之间的权衡。 截至目前，具有48个端口的1-Gbps以太网交换机基本上是一种商品组件，每个服务器连接一个机架的成本低于30美元/每Gbps。 因此，服务器机架内的带宽往往具有相同的特性。 然而，用于连接WSC集群的高端口数量的网络交换机具有完全不同的价格结构，比普通交换机贵十倍以上（每个1-Gbps端口）。 换句话说，具有10倍双向带宽的交换机的成本大约是100倍。 由于这种成本不连续性，WSC的网络结构通常组织成图1.1所示的两级层次结构。每个机架中的普通交换机通过少数几个上行链路向更昂贵的集群级交换机提供部分双向带宽用于机架间通信。

例如，一个机架上有40台服务器，每台服务器都有一个1-Gbps端口，可能会有四到八个1-Gbps上行链路连接到集群级交换机，对应着机架间通信的超额订阅因子在5到10之间。 在这样的网络中，程序员必须意识到相对稀缺的集群级带宽资源，并尽量利用机架级网络局部性，这增加了软件开发的复杂性，可能影响资源利用率。

或者，可以通过在互连网络上花费更多的钱来消除一些集群级网络瓶颈。 例如，Infiniband互连通常可以扩展到几千个端口，但每个端口的成本可能为500美元至2000美元。同样，一些网络供应商开始提供更大规模的以太网互连，但至少需要数百美元的成本。

每台服务器。 或者，可以通过构建“胖树”Clos网络来形成低成本的互连网络，使用普通的以太网交换机。 在网络和购买更多服务器或存储之间花费多少钱是一个应用特定的问题，没有一个单一正确的答案。 然而，目前我们将假设机架内的连接通常比机架间的连接更便宜。

1.6.3 存储层次结构

图1.2展示了一个典型WSC存储层次结构的程序员视图。一个服务器由多个处理器插槽组成，每个插槽都有一个多核CPU和其内部的缓存层次结构，本地共享和一些直接连接的磁盘驱动器。 机架内的DRAM和磁盘资源可以通过第一级机架交换机访问（假设有某种远程过程调用API），所有机架中的资源都可以通过集群级交换机访问。

1.6.4 量化延迟、带宽和容量

图1.3试图量化WSC的延迟、带宽和容量特性。为了说明，我们假设一个系统有2000台服务器，每台服务器有8GB的DRAM和四个1TB的磁盘驱动器。 每组40台服务器通过1Gbps的链路连接到机架级交换机，该交换机还有额外的八个1Gbps端口用于将机架连接到集群级交换机（一个

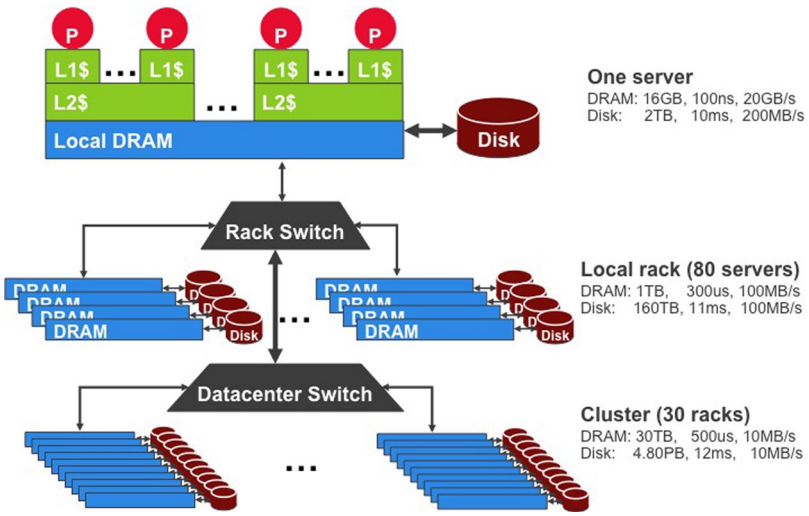

图1.2：WSC的存储层次结构。

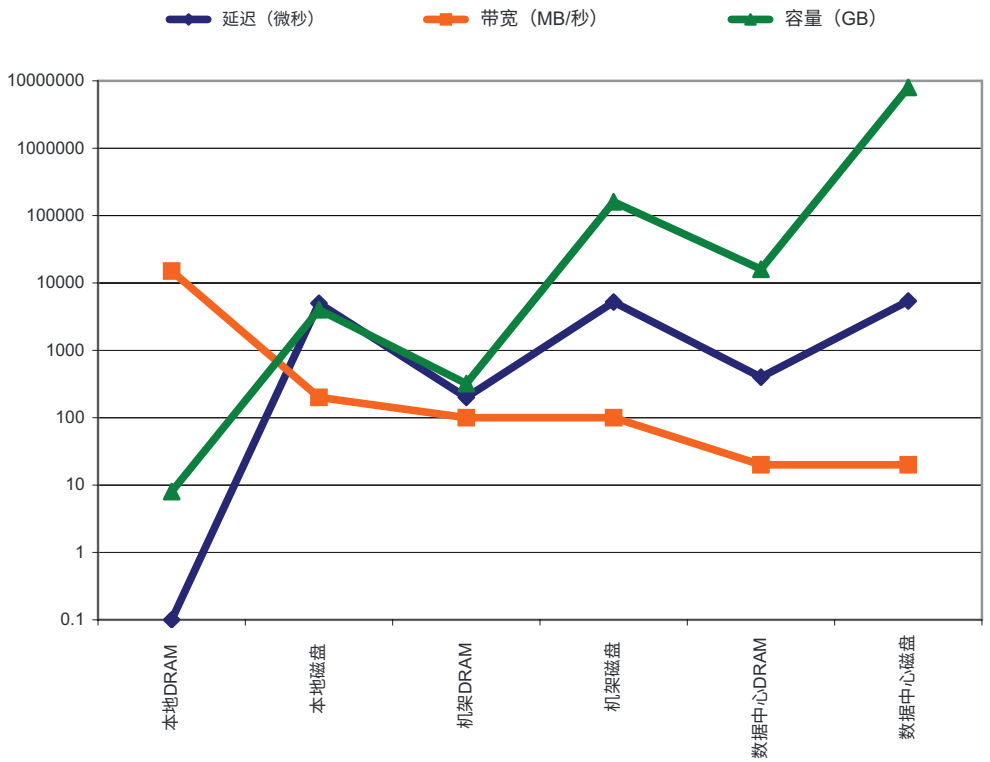

图**1.3**：WSC的延迟、带宽和容量。

超额订阅系数为5）。网络延迟数值假设使用基于套接字的TCP-IP传输，网络带宽数值假设每个位于超额订阅上行链路后面的服务器都使用其可用集群级带宽的公平份额。我们假设机架和集群级交换机本身没有内部超额订阅。对于磁盘，我们显示了典型的商品磁盘驱动器（SATA）延迟和传输速率。

该图显示了每个资源池的相对延迟、带宽和容量。 例如，本地磁盘的可用带宽为200 MB/s，而通过共享机架上行链路的离机磁盘的带宽仅为25 MB/s。 另一方面，集群中的总磁盘存储量几乎是本地DRAM的一千万倍。

一个需要比单个机架容纳更多服务器的大型应用程序必须有效地处理这些延迟、带宽和容量上的巨大差异。 这些差异比单台机器上的差异要大得多，使得在WSC上编程更加困难。

WSC架构师面临的一个关键挑战是以一种成本效益的方式平衡这些差异。 相反，软件架构师面临的一个关键挑战是构建集群基础设施和服务，将大部分复杂性隐藏在应用程序开发人员之前。

1.6.5 电力使用

能源和功耗在WSC设计中也是重要的考虑因素，因为如第5章中更详细讨论的那样，与能源相关的成本已成为这类系统总拥有成本的重要组成部分。 图1.4提供了一些关于现代IT设备中能源使用情况的见解，通过对2007年在Google部署的一代WSC的峰值功耗进行分类。

尽管这种分解可以根据系统如何配置对于给定的工作负载域而言会有很大的变化，但是图表表明，CPU不再是能源效率改进的唯一关注点，因为没有一个子系统主导整体能源使用情况。
第5章还讨论了电力传输和冷却方面的开销如何显著增加
WSC的实际能源使用量。

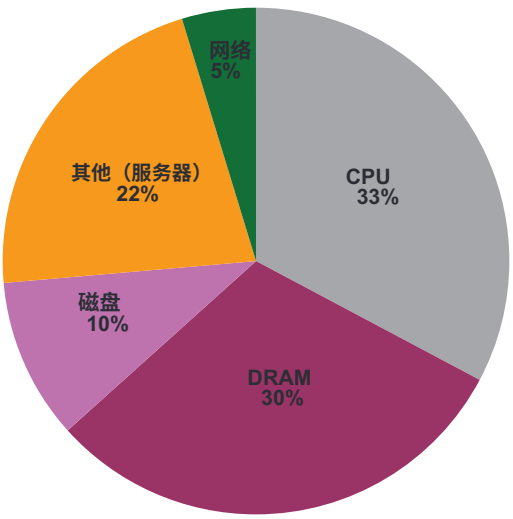

图1.4：谷歌其中一个数据中心（约2007年）硬件子系统峰值功耗的近似分布。

1.6.6 处理故障

WSC的规模之大要求互联网服务软件能够容忍相对较高的组件故障率。 例如，磁盘驱动器的年故障率可以高达4%以上[65，76]。 不同的部署报告了每年平均服务器级重启次数在1.2到16之间。由于如此高的组件故障率，跨数千台机器运行的应用程序可能需要每小时对故障条件做出反应。 我们将在第2章进一步扩展这个主题，描述应用领域，并在第7章讨论故障统计数据。

第二章

工作负载和软件基础设施

在仓库级计算机（WSCs）上运行的应用程序主导了许多系统设计的权衡决策。 本章概述了在大型互联网服务中运行的软件的一些独特特征，以及完整计算平台所需的系统软件和工具。 这里有一些术语，定义了典型WSC部署中的不同软件层次：

- 平台级软件- 通用固件、内核、操作系统分发和库，预期在所有单个服务器中存在，用于抽象单个机器的硬件并提供基本的服务器级服务。
- 集群级基础设施- 分布式系统软件的集合，用于管理资源并提供集群级服务；最终，我们将这些服务视为数据中心的操作系统。 例如，分布式文件系统、调度程序、远程过程调用（RPC）层，以及简化数据中心规模资源使用的编程模型，如MapReduce [19]、Dryad [47]、Hadoop [42]、Sawzall [64]、BigTable [13]、Dynamo [20]和Chubby [7]。
- 应用级软件- 实现特定服务的软件。将应用级软件进一步划分为在线服务和离线计算通常是有用的，因为它们往往具有不同的要求。 在线服务的例子包括谷歌搜索、Gmail和谷歌地图。离线计算通常用于大规模数据分析或作为生成在线服务中使用的数据的流水线的一部分；例如，构建网络索引或处理卫星图像以创建地图瓦片。

2.1 数据中心对比桌面电脑

互联网服务中的软件开发与传统的桌面/服务器模型有很多不同之处：

- 充足的并行性- 典型的互联网服务展示了大量的并行性，包括数据级别和请求级别的并行性。 通常，问题不在于找到

并行性，而是管理和有效利用应用程序中固有的显式并行性。 数据并行性源于需要处理的大量相对独立的记录集，例如数十亿个网页或数十亿个日志行。 这些非常大的数据集通常需要对每个并行（子）任务进行大量计算，从而帮助隐藏或容忍通信和同步开销。

同样，请求级并行性源于每秒接收到的数百或数千个请求，这是流行的互联网服务所接收到的。 这些请求很少涉及数据的读写共享或跨请求的同步。 例如，搜索请求基本上是独立的，并且处理的是一个主要是只读数据库；因此，计算可以在请求内部和不同请求之间容易地进行分区。 同样，虽然Web电子邮件事务确实修改用户数据，但来自不同用户的请求基本上是相互独立的，从而创建了数据分区和并发的自然单元。

- 工作负载变动- 互联网服务的用户通过相对明确定义和稳定的高级API（例如简单的URL）与服务的实现细节隔离，这使得快速部署新软件变得更加容易。 G*oogl*e服务的关键部分的发布周期大约为几周，而桌面软件产品的发布周期为几个月或几年。 例如，G*oogl*e的前端*W*eb服务器二进制文件每周发布一次，由数百名开发人员提交了近千个独立的代码更改- G*oogl*e搜索服务的核心几乎每2到3年重新实现一次。 这种环境为快速产品创新创造了重要的激励，但对于系统设计师来说，甚至从已建立的应用程序中提取有用的基准测试也很困难。

  此外，由于互联网服务仍然是一个相对较新的领域，新的产品和服务经常涌现，它们与用户的成功直接影响着数据中心中的工作负载组合。 例如，像YouTube这样的视频服务在相对短的时间内蓬勃发展，并且可能呈现出与数据中心中现有大型客户的计算周期需求非常不同的一组要求，从而可能以意想不到的方式影响WSC的最佳设计点。 这种积极的软件部署环境的一个有益的副作用是，硬件架构师不必为不可变的代码提供良好的性能负担。 相反，架构师可以考虑重写软件以利用新的硬件能力或设备的可能性。

- 平台的同质性- 数据中心通常比桌面作为软件开发目标平台更具同质性。 大型互联网服务运营商通常在任何给定时间内部署少量的硬件和系统软件配置。 主要是由于随着时间推移，部署更具成本效益的组件的激励而产生的显著异质性。 在一个平台世代内的同质性简化了集群级调度和负载平衡，并减少了

平台软件（内核、驱动程序等）的维护负担。同样，同质性可以使供应链更高效，修复过程更高效，因为自动和手动修复可以从对更少类型系统有更多经验中受益。

相比之下，桌面系统的软件可以对部署的硬件或软件平台做出很少的假设，它们的复杂性和性能特征可能因为需要支持成千上万甚至数百万的硬件和系统软件配置而受到影响。

- 无故障运行- 因为互联网服务应用程序在成千上万台机器的集群上运行 - 每台机器的可靠性并不比PC级硬件更高 - 单个故障率的乘法效应意味着每隔几个小时或更短的时间就会发生某种类型的故障（更多细节请参见第6章）。 因此，尽管对于桌面级软件来说，假设硬件故障无故障运行数月或数年可能是合理的，但对于数据中心级服务来说，这是不正确的 - 互联网服务需要在故障是日常生活的环境中工作。 理想情况下，集群级系统软件应该提供一个层次，将大部分复杂性隐藏在应用级软件之下，尽管对于所有类型的应用来说，实现这个目标可能是困难的。

尽管丰富的线程级并行性和更均匀的计算平台相对于桌面系统来说可以减少互联网服务中的软件开发复杂性，但规模、在硬件故障下运行的需求以及工作负载变化的速度则产生了相反的效果。

2.2 性能和可用性工具箱

由于在大规模部署中实现高性能或高可用性时具有广泛适用性，一些基本的编程概念在基础设施和应用层经常出现。 下表描述了一些最常见的概念。

	性能	可用性	描述
复制	是	是	数据复制是一种强大的技术，因为它可以同时提高性能和可用性。 当复制的数据不经常被修改时，它尤其强大，因为复制会使更新变得更加复杂。

（续）

	性能	可用性	描述
分片（分区）	是	是	将数据集分割成较小的片段（分片），并将它们分布在大量的机器上。 对数据集的操作被分派到托管分片的一些或所有机器上，并且结果由客户端合并。 分片策略可以根据空间限制和性能考虑而变化。 分片还有助于可用性，因为恢复小数据片段的速度比恢复大数据片段的速度更快。
负载平衡	是		在大规模服务中，服务级性能往往取决于数百或数千台服务器中最慢的响应者。 因此，减少响应时间的差异非常重要。 在分片服务中，可以通过偏向分片策略来平衡每个服务器的工作量来实现负载平衡。 该策略可能需要根据预期的请求混合或不同服务器的计算能力来确定。 请注意，即使是同质的机器，如果多个应用程序共享负载平衡服务器的子集，它们也可以向负载平衡客户端提供不同的性能特性。

（续）

性能	可用性	描述
		在复制服务中，负载平衡代理可以通过选择哪些服务器来调度新请求来动态调整负载。由于不同类型的请求所需的工作量并不总是恒定或可预测的，因此很难实现完美的负载平衡。
健康检查和看门狗定时器	是	在大规模系统中，故障通常表现为给定服务器的响应速度慢或无响应的行为。 在这种环境中，没有任何操作可以依赖于给定服务器的响应来取得进展。 此外，快速确定服务器是否过慢或无法访问，并将新请求引导到其他服务器是至关重要的。远程过程调用必须设置明智的超时值来中止长时间运行的请求，并且基础设施级软件可能需要不断检查通信服务器的连接级响应性并在需要时采取适当的措施。
完整性检查	是	在某些情况下，除了无响应-性外，故障还会表现为数据损坏。 虽然这些情况可能较少见，但确实会发生，并且通常以底层硬件或软件检查无法捕捉到的方式发生（例如，某些网络CRC检查的错误覆盖范围存在已知问题）。通过更改底层编码或添加更强大的冗余完整性检查，额外的软件检查可以缓解这些

（续）

	性能	可用性	描述
完整性检查		是	某些网络CRC检查的错误覆盖范围存在已知问题。通过更改底层编码或添加更强大的冗余完整性检查，额外的软件检查可以缓解这些问题。
应用程序-特定压缩	是		在现代数据中心中，设备成本的很大一部分是在各种存储层上。对于具有非常高吞吐量要求的服务来说，将尽可能多的工作集放入DRAM中非常关键；这使得压缩技术非常重要，因为解压缩的额外CPU开销仍然比访问磁盘的惩罚要低几个数量级。尽管通用的压缩算法在平均情况下表现得相当不错，但了解数据编码和值分布的应用级压缩方案可以实现显着优越的压缩因子或更好的解压缩速度。
最终一致性	是	是	通常，使用传统数据库管理系统提供的保证来保持多个副本的最新状态会显著增加复杂性，

（续）

性能	可用性	描述
		会影响分布式应用程序的性能，并降低可用性[90]。 幸运的是，许多应用程序类别对一致性要求较低，并且可以容忍有限时间内的不一致视图，只要系统最终返回到稳定一致的状态。

通过使用冗余计算技术，还可以提高大型并行应用程序的响应时间。 有几种情况可能导致大型并行作业的某个子任务比其兄弟任务慢得多，这可能是由于与其他工作负载或软件/硬件故障的性能干扰引起的。 由于明显的开销，冗余计算并没有像其他技术那样广泛部署。 然而，在某些情况下，一个大型作业的完成受到执行其一小部分子任务的影响。 其中一个例子是MapReduce论文中描述的慢工作者问题。 在这种情况下，一个较慢的工作者可以决定一个巨大并行任务的响应时间。 MapReduce的策略是在作业结束时识别这种情况，并仅为那些较慢的作业启动冗余工作者。 这种策略会增加资源使用率几个百分点，同时将并行计算的完成时间缩短超过30%。

2.3 集群级基础设施软件

就像在单台计算机中需要操作系统层来管理资源和提供基本服务一样，在由成千上万台计算机、网络和存储组成的系统中，也需要一层软件在更大的规模上提供类似的功能。 我们将这一层称为集群级基础设施。 接下来的段落描述了构成这一层的四个广泛的基础设施软件组。

2.3.1 资源管理

这可能是集群级基础设施层中最不可或缺的组件。 它控制用户任务与硬件资源的映射，强制执行优先级和配额，并提供基本的任务管理服务。 在其最简单的形式中，它可以简单地是一个接口，用于手动（和静态地）将一组机器分配给特定的用户或作业。更有用的版本应该提供更高级别的抽象，自动分配资源，并允许在更细粒度上共享资源。 这种系统的用户应该能够以相对较高的级别指定其作业要求（例如，需要多少CPU性能、内存容量、网络带宽等），并且调度程序将其转换为适当的资源分配。 在进行调度决策时，集群调度程序越来越重要的是考虑功率限制和能源使用优化，不仅要处理紧急情况（如冷却设备故障），还要最大限度地利用预配的数据中心功率预算。 第5章对此主题提供了更详细的信息。

2.3.2 硬件抽象和其他基本服务

几乎每个大规模分布式应用都需要一小组基本功能。 例子包括可靠的分布式存储、消息传递和集群级同步。 在大型集群中，正确实现这种类型的功能并具有高性能和高可用性是复杂的。 明智的做法是避免为每个应用重新实现这种棘手的代码，而是创建可以重用的模块或服务。 GFS [31]、Dynamo [20]和Chubby [7]是Google和Amazon开发的可靠存储和锁定服务的示例。

2.3.3 部署和维护

在小规模部署中可以手动处理的许多任务在大规模系统中需要大量基础设施来实现高效运作。 例子包括软件映像分发和配置管理、监视服务性能和质量以及在紧急情况下为操作员处理警报。 微软的Autopilot系统[48]为Windows Live数据中心的某些功能提供了一个示例设计。 监控硬件设备群的整体健康状况还需要仔细监控、自动诊断和修复工作流的自动化。 Google的系统健康基础设施，在Pinheiro等人的文章[65]中有所描述，是实现高效健康管理所需的软件基础设施的一个示例。 最后，在这种规模的系统中进行性能调试和优化也需要专门的解决方案。

在加州大学伯克利分校开发的X-Trace [30]系统是一个监控基础设施的例子，旨在对大型分布式系统进行性能调试。

2.3.4 编程框架

在前面的段落中描述的整个基础设施简化了硬件资源的部署和高效使用，但它并没有从根本上隐藏大型硬件集群作为普通程序员的目标的固有复杂性。 从程序员的角度来看，硬件集群具有深层复杂的内存/存储层次结构、异构组件、容易故障的组件以及某些资源的稀缺性（如DRAM和数据中心级网络带宽）。 解决的问题规模显然会进一步复杂化。 在大规模服务中，某些类型的操作或问题子集是足够常见的，因此构建针对性的编程框架可以极大地简化新产品的开发。 MapReduce [19]、BigTable [13]和Dynamo [20]是基础设施软件的良好示例，它们通过自动处理数据分区、分布和容错性在各自领域内极大地提高了程序员的生产力。

2.4 应用级软件

网络搜索是最早获得广泛流行的大规模互联网服务之一，随着网络内容在九十年代中期爆炸式增长，组织这么大量的信息已经超出了当时现有的人工管理目录服务的能力。 然而，随着家庭和企业的网络连接不断改善，通过互联网提供新服务变得更具吸引力，有时会取代传统上存在于客户端的计算能力。 基于Web的地图和电子邮件服务是这些趋势的早期例子。 提供的服务范围的增加导致了更多种类的应用级需求。 例如，搜索工作负载可能不需要具备高性能原子更新能力的基础设施，并且对硬件故障有一定的容错性（因为在网络搜索中绝对精确度并不是那么关键）。 但对于跟踪用户点击赞助链接（广告）的应用程序来说，情况就不同了。 对广告点击来说，基本上是小额财务交易，需要满足事务性数据库管理系统所期望的许多保证。

一旦考虑到多个服务的不同需求，就会清楚地认识到数据中心必须是一个通用的计算系统。 尽管专门的硬件解决方案可能适用于服务的个别部分，但需求的广泛性使得专门的硬件在整体运作中产生的影响较小的可能性更大。 另一个不利于硬件专门化的因素是工作负载变化的速度；产品需求迅速演变，聪明的程序员会从经验中学习并重新编写基准算法和数据结构。

而硬件本身的演进速度远远赶不上。 因此，专门的硬件解决方案在实施时存在相当大的风险，甚至对于其设计的问题领域来说，它可能已经不再适用。

2.4.1 工作负载示例

我们在这里的目标不是详细描述互联网服务的工作负载，特别是因为这个市场的动态性将使得这些描述在发布时已经过时。 然而，描述两种广泛应用类别的高级工作负载是有用的：在线服务和批处理（离线）处理系统。 在这里，我们概述了一个Web搜索应用程序的基本架构，作为在线系统的示例，以及使用MapReduce的基于引文相似性计算作为批处理工作负载的示例。

2.4.2 在线：网络搜索

这是一个典型的“大海捞针”问题。 虽然准确确定Web的大小是困难的，但可以肯定的是它由数千亿个独立的文档组成，并且不断增长。 如果我们假设Web包含1000亿个文档，每个文档的平均大小为4 kB（压缩后），那么这个大海捞针的问题大约是400 TB。Web搜索的数据库是通过将这个文档集合反转来构建的，创建一个以图2.1所示的逻辑格式的存储库。一个词典结构将每个词与存储库中的每个术语关联起来。 termID标识了包含该术语的文档列表，以及有关该术语的一些上下文信息，例如位置和其他各种属性（例如，术语是否在文档标题中）。

生成的倒排索引的大小取决于具体的实现方式，但通常与原始存储库的数量级相同。 典型的搜索查询由一系列术语组成，系统的任务是找到包含所有术语的文档（AND查询），并确定其中哪些文档最有可能满足用户的需求。 查询可以选择性地包含特殊运算符，以指示交替（OR运算符）或限制搜索仅限于特定顺序的术语出现（短语运算符）。 为了简洁起见，我们重点关注更常见的AND查询。

考虑一个查询，比如[纽约餐馆]。 搜索算法必须遍历每个术语（new，york，restaurant）的倒排列表，直到找到包含在所有三个倒排列表中的所有文档。 在那时，它使用各种参数对找到的文档进行排名，例如文档的整体重要性（在Google的情况下，这将是PageRank分数[59]）以及与文档中术语出现相关的许多其他属性（例如出现次数、位置等），然后将排名最高的文档返回给用户。

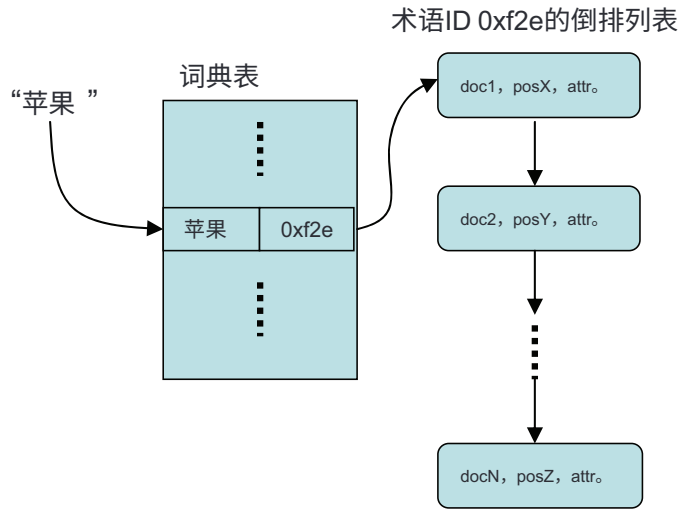

图2.1：Web索引的逻辑视图。

例如出现次数、位置等），然后将排名最高的文档返回给用户。

鉴于索引的庞大规模，这个搜索算法可能需要在几千台机器上运行。 这是通过将索引分割成负载均衡的子文件并将它们分布在所有机器上来实现的。 索引分区可以按文档或按术语进行。

用户查询由前端Web服务器接收并分发到索引集群中的所有机器。 根据吞吐量或容错性的需要，索引子文件的多个副本可以放置在不同的机器上，这种情况下只有一部分机器参与到给定的查询中。 索引服务机器计算本地结果，对其进行预排序，并将最佳结果发送到前端系统（或某个中间服务器），该系统从整个集群中选择最佳结果。 此时，只知道与生成的网页命中相对应的doc_IDs列表。 需要第二阶段来计算实际的标题、URL和与搜索词相关的特定查询文档片段。 通过将doc_IDs列表发送到包含文档副本的一组机器来实现这个阶段。

再次，考虑到存储库的规模，需要将其分区并放置在大量的服务器上。

上述操作的总用户感知延迟需要是秒的一小部分；因此，这种架构非常注重降低延迟。然而，

高吞吐量也是一个关键的性能指标，因为一个热门服务可能需要支持每秒数千个查询。 索引经常更新，但在处理单个查询的时间粒度内，可以将其视为只读结构。 此外，由于除了最后的合并步骤之外，不需要不同机器之间进行索引查找的通信，计算非常高效地并行化。 最后，通过利用不同Web搜索查询之间没有逻辑交互的事实，可以进一步实现并行化。

如果索引按照doc_ID进行分区（分片），这个工作负载在平均带宽方面对网络需求相对较小，因为机器之间交换的数据量通常不会比查询本身的大小大很多（大约一百字节左右），但会表现出一些突发行为。 基本上，前端的服务器充当流量放大器，它们将单个查询分发给非常多的服务器。 这会在请求路径上产生一阵流量爆发，可能还会在响应路径上产生流量爆发。 因此，即使整体网络利用率较低，仍需要对网络流量进行仔细管理，以最小化数据包丢失。

最后，由于Web搜索是一项在线服务，它会受到正常的流量变化的影响，因为用户在一天的不同时间更活跃于Web上。图2.2展示了这种影响，显示在高峰使用时间，流量可能是非高峰时期的两倍以上。
这种变化对系统操作员来说是一个挑战，因为服务必须为比平均行为强烈得多的流量强度进行调整。

2.4.3 离线：学术文章相似性

用户请求提供了许多大规模计算的示例，这些计算是互联网服务运营所需的。 这些计算通常是需要进行数据并行工作负载的类型，以准备或打包随后在在线服务中使用的数据。 例如，计算PageRank或从Web存储库创建倒排索引文件属于这一类别。 但在这里，我们使用了一个不同的例子：在学术论文和期刊的存储库中查找相似的文章。

这对于提供科学出版物访问的互联网服务非常有用，例如Google Scholar（http://scholar.google.com）。文章相似性关系是关键字搜索系统之外的另一种找到相关信息的方式；在找到感兴趣的文章后，用户可以要求服务显示与原始文章强相关的其他文章。

有几种计算相似度得分的方法，通常适合使用多种方法并结合结果。 在学术文章中，已经有各种形式的引用分析。

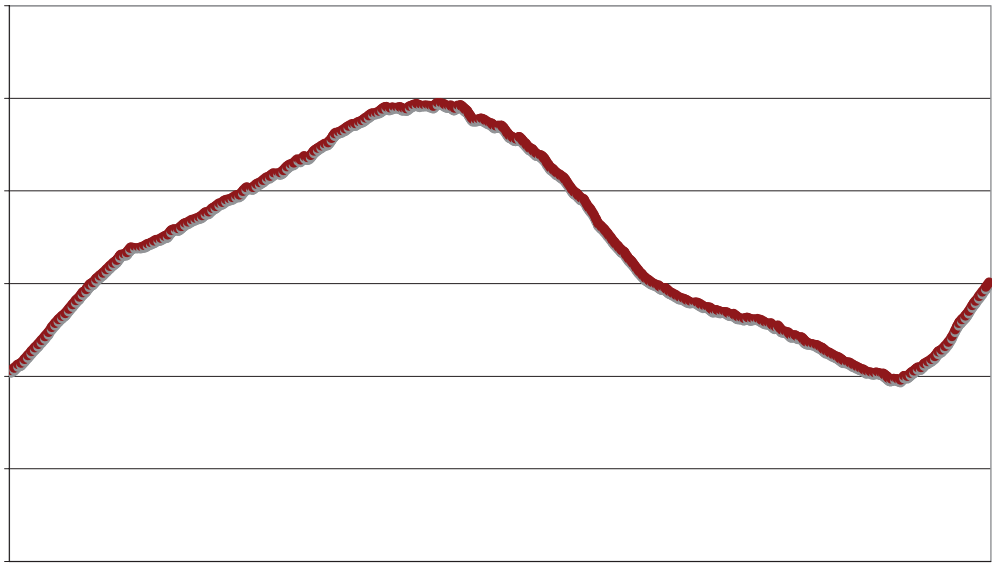

图2.2：一个数据中心中搜索服务每日流量波动的示例；*x*轴表示24小时周期，*y*轴表示每秒查询量。

已知提供高质量的相似度得分。 在这里，我们考虑一种称为共引分析的分析类型。 其基本思想是将引用文章A和B的每篇文章都视为A和B之间相似度的一票。在对所有文章进行适当归一化后，我们得到了所有文章对之间（共引）相似度的数值分数，并创建了一个数据结构，该结构可以为每篇文章返回一个按共引分数排序的相似文章列表。 这个数据结构会定期更新，每次更新都成为在线服务的一部分。

计算从引用图开始，该图将每篇文章的标识符映射到引用它的一组文章。输入数据被分成数百个大小相近的文件（例如，可以通过对文章标识符进行指纹处理，将其除以输入文件的数量，并使用余数作为文件ID来完成），以实现高效的并行执行。我们使用一系列的MapReduce运行来获取引用图并为所有文章生成共引相似度得分向量。 在第一个Map阶段，我们获取每个引用列表（A1，A2，A3，...，A*n*）并生成引用列表中的所有文档对，将它们传递给Reduce阶段，该阶段计算每对的出现次数。 这个第一步的结果是一个将所有共引文档对与共引计数相关联的结构。 请注意，这远远小于二次方程。

由于大多数文档的共引用计数为零，因此它们被省略。 第二个MapReduce过程将给定文档的所有条目分组，对它们的分数进行归一化，并生成一个按相似性分数递减排序的文档列表。

这个两阶段的数据并行程序在数百台服务器上执行，每个阶段的计算相对较轻，但在Map和Reduce工作节点之间进行了大量的全对全通信。 然而，与Web搜索不同，网络流量是连续的，这使得它更适合现有的拥塞控制算法。 与Web搜索相反，单个任务的延迟并不那么重要，比起工作负载的整体并行效率。

2.5 监控基础设施

集群级基础设施软件层的一个重要部分涉及各种形式的系统内省。 由于工作负载和硬件基础设施的规模和复杂性，使得监控框架成为任何此类部署的基本组成部分，我们在这里对其进行更详细的描述。

2.5.1 服务级仪表盘

系统操作员必须跟踪互联网服务是否达到目标服务水平。

监控信息必须非常及时，以便操作员（或自动化系统）能够快速采取纠正措施，避免重大中断-在几秒钟内，而不是几分钟内。 幸运的是，最关键的信息仅限于可以从前端服务器收集的少数信号，例如用户请求的延迟和吞吐量统计数据。 在其最简单的形式中，这样的监控系统可以简单地是一个脚本，每隔几秒钟轮询所有前端服务器以获取适当的信号，并在仪表板上显示给操作员。

大规模服务通常需要更复杂和可扩展的监控支持，因为前端的数量可能非常大，并且需要更多的信号来描述服务的健康状况。 例如，收集信号本身以及它们随时间的变化可能非常重要。 除了延迟和吞吐量之外，系统还可能需要监控其他业务特定的参数。 监控系统可能需要支持一个简单的语言，让操作员基于正在监控的基准信号创建派生参数。 最后，系统可能需要根据监控值和阈值向值班操作员生成自动警报。 调整好警报系统可能会有些棘手，因为由于误报而频繁触发的警报会导致操作员忽视真正的警报，而只在极端情况下触发的警报可能会让操作员太晚注意到并导致无法顺利解决潜在问题。

2.5.2 性能调试工具

尽管服务级别仪表板可以让操作员快速识别服务级别问题，但它们通常缺乏详细信息，以了解服务为什么变慢或者未能满足要求的原因。 操作员和服务设计师都需要工具来帮助他们理解许多程序之间的复杂交互，这些程序可能在数百台服务器上运行，以便他们可以确定性能异常的根本原因并识别瓶颈。 与服务级别仪表板不同，性能调试工具可能不需要实时生成信息以进行在线操作。 可以将其视为数据中心中类似于CPU分析器的工具，它确定了程序中耗时最长的函数调用。

已经提出了分布式系统跟踪工具来解决这个需求。 这些工具试图确定在分布式系统中代表给定发起者（如用户请求）完成的所有工作，并详细说明各个组件之间的因果或时间关系。

这些工具通常分为两大类：黑盒监控系统和应用程序/中间件仪器系统。 WAP5 [72]和Sherlock系统[8]是黑盒监控工具的例子。 他们的方法是观察系统组件之间的网络流量，并通过统计推理方法推断因果关系。 因为它们将所有系统组件（除了网络接口）视为黑盒，所以这些方法的优点是不需要了解或依赖应用程序或软件基础设施组件的帮助。 然而，这种方法本质上牺牲了信息准确性，因为所有关系都必须通过统计推断得出。 收集和分析更多的消息数据可以提高准确性，但会增加监控开销。

基于仪器的追踪方案，如Pip [71]，Magpie [9]和X-trace [30]，利用了显式修改应用程序或中间件库以在机器之间和机器内部模块边界传递追踪信息的能力。 注释模块通常还会将追踪信息记录到本地磁盘，以供外部性能分析程序后续收集。 这些系统可以非常准确，因为不需要推断，但需要对分布式系统的所有组件进行仪器化以收集全面的数据。 Google开发的Dapper系统是一种基于注释的追踪工具的例子，通过仪器化与所有应用程序常见链接的几个关键模块（如消息传递、控制流和线程库），它在应用程序级别上保持了有效的透明性。

最后，将能力构建到二进制文件（或运行时系统）中以获取生产中程序的CPU、内存和锁争用概况非常有用。 这可以消除重新部署新的二进制文件来调查性能问题的需要。

2.5.3 平台级监控

分布式系统跟踪工具和服务级别仪表板都在测量应用程序的健康和性能。 这些工具可以推断硬件组件可能出现故障，但这仍然是一种间接评估。 此外，由于集群级基础设施和应用程序级软件都设计成能容忍硬件组件故障，因此在这些级别上的监控可能会错过大量潜在的硬件问题，使其逐渐积累，直到软件容错能力无法再减轻它们。 在那时，服务中断可能会非常严重。 需要持续且直接监控计算平台健康状况的工具来理解和分析硬件和系统软件故障。 在第6章中，我们将更详细地讨论一些这些工具及其在谷歌基础设施中的使用。

2.6 BUy VS. BUILD

传统的IT基础设施广泛使用第三方软件组件，如数据库和系统管理软件，并专注于创建特定于特定业务的软件，在产品提供的业务价值上增加，例如作为应用服务器和数据库引擎之上的业务逻辑。 大型互联网服务提供商，如谷歌，通常采用不同的方法，即应用特定逻辑和大部分集群级基础设施软件都是自行编写的。 平台级软件确实使用第三方组件，但这些组件往往是可以根据需要在内部进行修改的开源代码。 因此，整个软件堆栈更多地受到服务开发人员的控制。

这种方法增加了重要的软件开发和维护工作，但可以在灵活性和成本效益方面提供重要的好处。 灵活性在必须解决关键功能或性能错误时非常重要，可以在所有层面上快速修复错误。 在面对复杂的系统问题时，这也非常有优势，因为它提供了多种解决方案。 例如，不希望的网络行为可能在应用程序层面上很难解决，但在RPC库层面上相对简单，或者反过来。

从历史上看，支持建设而非购买的主要原因是所需的仓库规模软件基础设施在商业上并不可用。 此外，第三方软件提供商可能很难充分测试和调整他们的软件，除非他们自己维护大型集群。 最后，内部软件可能更简单更快，因为它可以设计来满足一小部分服务的需求，因此在该领域可以更加高效。 例如，BigTable省略了传统SQL数据的一些核心功能。

为了实现其预期的使用情况，BigTable通过放弃一些核心功能来获得更高的吞吐量和可扩展性，而GFS由于类似的原因无法提供完全符合Posix标准的文件系统。

2.7 进一步阅读

Hamilton [43]、Brewer [10]和Vogels [90]的文章提供了关于如何在非常大规模上部署互联网服务的一般问题的有趣进一步阅读。

第三章

硬件构建模块

正如前面提到的，仓库级计算机（WSCs）的架构主要由其构建模块的选择来定义。 这个过程类似于选择逻辑元素来实现微处理器，或者在设计服务器平台时选择正确的芯片组和组件。 在这种情况下，主要的构建模块是服务器硬件、网络结构和存储层次结构组件。 在本章中，我们重点关注服务器硬件的选择，目的是培养对这些选择的直觉。 我们希望在本出版物的即将发布的修订版中，通过额外的材料来扩展这一部分，涵盖存储和网络方面的内容。

3.1 成本效益的硬件

低端服务器集群是WSCs今天的首选构建模块[5]。 这是因为低端服务器的成本效益要高于之前一直是首选构建高性能和技术计算空间的高端共享内存系统。 低端服务器平台与大规模个人计算市场共享许多关键组件，因此更能从规模经济中获益。

由于价格波动和性能受基准特性和基准测试投入的努力水平的影响，往往很难进行有意义的成本效益比较。 TPC-C基准测试[85]的数据可能是最接近同时包含硬件成本和应用层性能的信息的一组指标。 因此，在这个案例中，我们将2007年末最佳性能的TPC-C系统（HP Integrity Superdome-Itanium2 [87]）与性价比类别中的顶级系统（HP ProLiant ML350 G5 [88]）进行了比较。 这两个系统在一个月内进行了基准测试，TPC-C执行摘要中包含了两个平台的成本细分，以便我们可以进行成本效益的合理分析。

表3.1显示了这两台服务器的基本机器配置。 根据官方基准测试指标，ProLiant的成本效益约为Superdome的四倍。这在选择平台时具有足够的意义。 TPC-C基准测试的扩展规则在官方指标中可能对ProLiant造成了一定的惩罚，因为它要求相当大的存储空间。

表3.1：服务器硬件配置、基准测试结果和衍生（非官方）成本效益指标，适用于大型SMP服务器和低端PC级服务器。

	HP INTEGrITy SUPEr Do ME-ITANIUM2	HP ProLIANT ML350 G5
处理器	64个插槽，128个核心（双线程），1.6 GHzItanium2，12 MB末级缓存	1个插槽，四核心，2.66 GHz X5355 CPU, 8 MB末级缓存
内存	2,048 GB	24 GB
磁盘存储	320,974 GB，7,056个驱动器	3,961 GB，105个驱动器
TPC-C价格/性能	$2.93/tpmC	$0.73/tpmC
价格/性能（仅限服务器硬件）	$1.28/每分钟交易量	$0.10/每分钟交易量
价格/性能（仅限服务器硬件）（无折扣）	$2.39/每分钟交易量	$0.12/每分钟交易量

在官方基准配置中，子系统的存储子系统占总服务器硬件成本的四分之三，而在Superdome配置中约为40%。 如果我们排除存储成本，ProLiant平台的价格/性能优势将增加3倍以上。 基准测试规则还允许将典型的硬件折扣应用于价格/性能指标中使用的总成本，这再次使Superdome受益，因为它是一个更昂贵的系统（约1200万美元对比ProLiant的75000美元），因此可以享受更深的折扣。 假设我们有相同的预算来购买这两种系统，假设对于任何一种情况都有相同的折扣水平可能是合理的。 如果在这两种情况下都取消折扣，ProLiant服务器硬件的成本效益将大约是Superdome的20倍。

3.1.1 并行应用性能如何？

上述分析对高端服务器来说确实不公平，因为它没有考虑到其极高的互联性能。 大型SMP中的节点可以进行通信

在延迟约为100纳秒的情况下，通常部署在集群中的基于局域网的网络将经历100毫秒或以上的延迟。无疑，具有密集通信模式的工作负载在128个处理器核心的SMP中的性能要明显优于32个四核低端服务器的以太网连接集群。 然而，在WSC环境中，即使是最大的高端SMP服务器也可能无法容纳如此大的工作负载。 因此，有趣的问题是，当使用一类机器构建成千上万个处理器核心的集群时，它们的性能会有多大提升。 下面这个非常简单的模型可以帮助我们从高层次上理解这样的比较。

假设给定的并行任务执行时间可以粗略地建模为固定的本地计算时间加上对全局数据结构的访问延迟惩罚。 如果计算适合于单个大型共享内存系统，那么这些全局数据访问将以大约DRAM速度（~100 ns）执行。 如果计算只适合于多个这样的节点，一些全局访问将会慢得多，大约是典型的局域网速度（~100 µs）。 让我们进一步假设对全局存储的访问在所有节点之间均匀分布，以便映射到本地节点的全局访问的比例与系统中节点数量成反比。这里的节点是共享内存域，例如一个Integrity系统或一个ProLiant服务器。

如果固定的本地计算时间为1毫秒的数量级-对于高吞吐量的互联网服务来说是一个合理的值-则确定程序执行时间的方程如下：

$$\text{执行时间} = 1\text{毫秒} + f * [100\ \text{ns}/\text{节点数} + 100\ \mu\text{s} * (1 - 1/\text{节点数})]$$

其中变量 f是每个（1毫秒）工作单元的全局访问次数。 在图3.1中，我们绘制了并行工作负载的执行时间，随着参与计算的节点数量的增加而增加。 对于不同的 f值，显示了三条曲线，表示具有轻通信（f=1）、中等通信（f=10）和高通信（f=100）模式的工作负载。 请注意，在我们的模型中，节点数量越大，远程全局访问的比例越高。

图3.1中的曲线具有两个值得强调的有趣方面。 首先，在轻通信情况下，使用多个节点的集群相对性能下降较小。 对于中等和高通信模式，惩罚可能相当严重，但从一个节点到两个节点时，惩罚最为显著，随着集群规模的增加，额外惩罚逐渐减少。 使用这个模型，单个128处理器SMP相对于32个4处理器SMP的集群的性能优势可能超过10倍×。

根据定义，WSC系统将由数千个处理器核心组成。 因此，我们希望使用这个模型来比较使用类似Superdome服务器构建的集群与使用类似ProLiant服务器构建的集群的性能。 在这里，我们假设每个核心的性能是相同的。

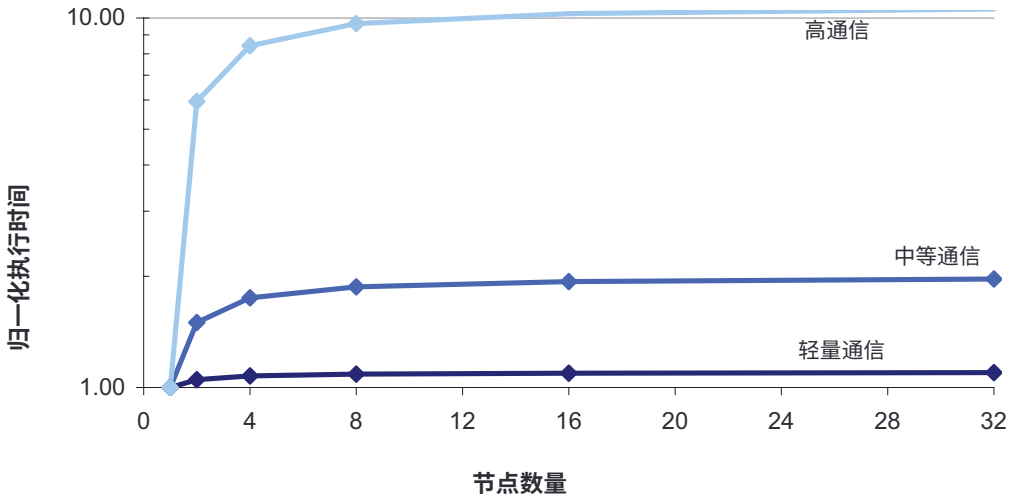

图**3.1**：并行任务的执行时间随SMP节点数量增加而变化，对应三个不同的通信强度水平。 执行时间经过归一化处理，并以对数比例尺绘制。

对于两个系统，服务器之间使用以太网级别的互连结构。 我们相信，尽管我们的模型非常简单（例如，它没有考虑争用效应），但足以捕捉我们感兴趣的效应。

在图3.2中，我们将我们的模型应用于大小在512到4,192个核心之间变化的集群，并展示了使用Superdome类型服务器（单个共享内存域中的128个核心）与使用ProLiant类型服务器（四核SMP）的实现之间的性能优势。在图中，512个核心的集群将四个Superdome类型系统的性能与使用128个ProLiant类型系统构建的集群的性能进行比较。 有趣的是，随着集群规模的增加，基于高端服务器的集群的性能优势迅速减弱。 如果应用程序需要超过两千个核心，那么一个由512个低端服务器构建的集群在重度通信模式下的性能与一个由16个高端服务器构建的集群相差不到5%。 在这种性能差距下，高端服务器的价格溢价（4-20倍更高）使其成为一个不具吸引力的选择。

这个分析的重点是定性的。 它主要旨在说明当为应用程序设计系统时，我们需要以不同的方式进行基准平台选择。这些应用程序太大，无法在任何单个高端服务器上运行。总的观点是，在仓库规模上对系统有益的性能效果最为重要。 对于局部计算的性能增强（例如快速SMP风格的通信），仍然非常重要。但是，如果它们带来了巨大的额外成本，它们的成本效益可能不如小型计算机适用于WSCs。

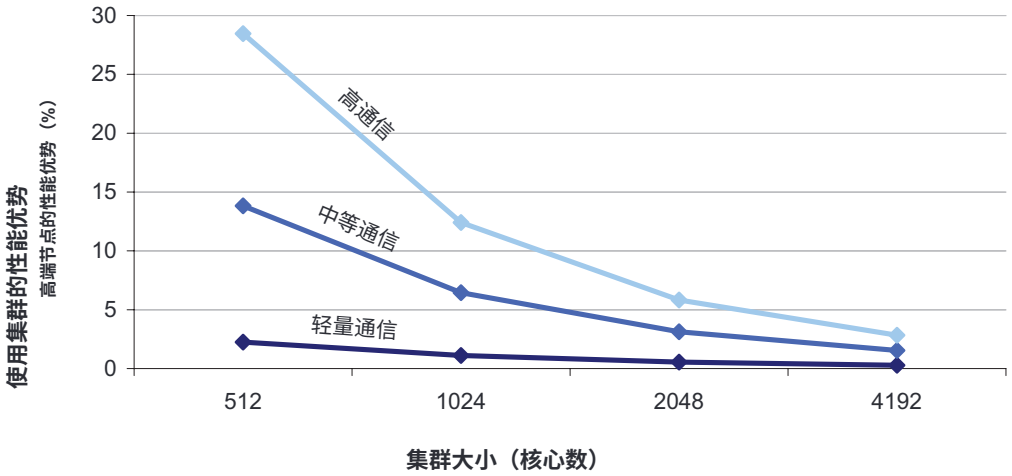

图3.2：使用高端服务器节点（128核SMP）构建的集群与使用低端服务器节点（四核SMP）构建的具有相同处理器核心数量的集群相比，性能优势的变化。

例如，我们的例子中的通信仍然非常重要。但是，如果它们带来了巨大的额外成本，它们的成本效益可能不如小型计算机适用于WSCs。

3.1.2 你能走多低端?

显然，可以进一步扩展上述论点，使用嵌入式组件构建的计算构建模块更小。 例如，Lim等人[51]认为，在考虑了所有与功耗相关的成本（包括数据中心建设成本的摊销和能源成本）之后，这些替代方案对于低端服务器平台是有优势的。 最近，Hamilton [44]提出了类似的论点，尽管使用的是PC级组件而不是嵌入式组件。 使用较小、较慢的CPU的优点与使用中档商品服务器而不是高端SMP的论点非常相似：

- 中档服务器上的多核CPU通常比低端处理器具有更高的性价比，因此可以用多个较小的CPU以更低的价格购买相同数量的吞吐量。
- 许多应用程序受内存限制，因此更快的CPU对于大型应用程序的扩展性不好，进一步增强了较简单CPU的价格优势。

- 较慢的CPU更具功耗效率；通常，当CPU频率降低k倍时，CPU功耗减少$O(k^2)$。

然而，抵消效应减弱了这些优势，使得越来越小的构建模块对于WSCs越来越不具吸引力。

也许最重要的是，尽管许多互联网服务受益于看似无限的请求和数据级并行性，但这些系统并不免疫于阿姆达尔定律。随着提供的并行线程数量的增加，减少串行化和通信开销可能变得越来越困难。 在极限情况下，由一个极慢的单线程硬件执行的固有串行工作量将主导整体执行时间。 此外，处理并行化请求的线程数量越多，所有这些并行任务的响应时间变化就越大，因为负载平衡变得更加困难，不可预测的性能干扰变得更加明显。 因为通常所有并行任务必须在请求完成之前完成，所以整体响应时间变为任何子任务的最大响应时间，并且更多的子任务将进一步推进子任务响应时间的长尾，从而影响整体服务延迟。

因此，尽管硬件成本可能降低，但软件开发成本可能会增加，因为必须显式地并行化或进一步优化更多的应用程序。 例如，假设一个网络服务当前每个用户请求的延迟为1秒，其中一半是由CPU时间引起的。 如果我们切换到性能较慢三倍的低端服务器集群，服务的响应时间将增加到2秒，并且应用程序开发人员可能需要花费大量的精力来优化代码，以恢复到1秒的延迟水平。

随着更多数量的较小系统，网络需求也增加，增加了网络延迟和网络成本（因为已经有了更多的端口在一个已经昂贵的交换结构中）。 通过本地互连少量较慢的服务器来共享网络链接可以减轻这种影响，但是这种互连的成本可能抵消了切换到更便宜的CPU所获得的价格优势。

较小的服务器可能导致利用率降低。 将一组应用程序分配到服务器池中作为一个装箱问题来考虑——每个服务器都是一个箱子，我们尽量将尽可能多的应用程序放入每个箱子中。显然，当箱子较小时，这个任务变得更加困难，因为许多应用程序可能无法完全填满一个服务器，但却使用了太多的CPU或RAM，以至于不允许第二个应用程序与之共存在同一台服务器上。

最后，即使是尴尬并行算法有时在计算和数据被分割成较小的部分时也本质上不太高效。 例如，当并行计算的停止准则基于全局信息时，就会发生这种情况。 为了避免昂贵的全局通信和全局锁争用，本地任务可以使用基于它们本地的启发式方法。

仅仅是进展，这样的启发式方法自然更加保守。 因此，当地的子任务可能会执行更长的时间，如果有关全局进展的提示更好的话，它们可能会执行得更好。 当这些计算被分割成较小的部分时，这种开销往往会增加。

作为一个经验法则，低端服务器构建模块必须比高端替代品具有更好的成本效益才能具有竞争力。 目前，许多大规模服务的最佳选择似乎在服务器级机器的低端范围内（与高端个人计算机的范围有所重叠）。

3.1.3 平衡设计

计算机架构师接受培训，解决从构成WSC的各种构建模块中找到性能和容量的正确组合的问题。在本章中，我们展示了一个例子，说明只有当人们关注整个WSC系统时，才能看到正确的构建模块。 在这个层面上，也必须解决平衡的问题。 重要的是要对将在系统上执行的工作负载的各种资源消耗进行特征化，同时牢记三个重要的考虑因素：

- 聪明的程序员可能能够重组他们的算法，以更好地匹配一个更便宜的设计替代方案。 在软件-硬件协同设计的过程中，有机会找到解决方案，但要注意不要设计出过于复杂的机器，以免难以编程。

- 对于硬件来说，最具成本效益和平衡的配置可能是与多个工作负载的综合资源需求相匹配，而不一定完全适合任何一个工作负载。 例如，一个寻址受限的应用程序可能无法充分利用非常大的磁盘驱动器的容量，但可以与需要主要用于归档目的的应用程序共享该空间。

- 可互换的资源往往更有效地利用。 只要在一个数据中心内有合理的连接性，就应该努力创建能够灵活利用远程服务器资源的软件系统。 这在许多方面影响着平衡决策。 例如，有效地使用远程磁盘驱动器可能要求服务器的网络带宽等于或高于所有本地连接到服务器的磁盘驱动器的峰值带宽之和。

正确的设计点取决于工作负载本身的高级结构以外的因素，因为数据大小和服务的流行度也起着重要作用。 例如，一个具有大量数据集但请求流量相对较小的服务可能能够直接从磁盘驱动器中提供大部分内容，因为存储成本低（以每GB美元计）但吞吐量低。极受欢迎的服务

要么具有较小的数据集大小，要么具有显著的数据局部性以便利用内存服务。 最后，这个领域的工作负载变化也是WSC架构师面临的挑战。

软件基础可能发展得如此之快，以至于服务器设计选择在其寿命期间变得次优（通常为3-4年）。 对于整个WSC来说，这个问题更加重要，因为数据中心设施的寿命通常跨越多个服务器寿命，或者超过十年左右。在这些情况下，有必要设想WSC系统在其寿命期间可能需要的机器或设施升级，并在设计阶段考虑这一点。

第四章

数据中心基础知识

数据中心本质上是消耗电力并产生热量的大型设备。 数据中心的冷却系统会移除这些热量，但在这个过程中会消耗额外的能源，而这些热量也必须被移除。 因此，不足为奇的是，数据中心的建设成本主要与所提供的电力量和需要移除的热量成正比；换句话说，大部分资金要么用于电力调节和分配，要么用于冷却系统。 大型数据中心的典型建设成本在每瓦10-20美元的范围内（参见第6.1节），但根据规模、位置和设计的不同而有很大的变化。

4.1 数据中心的等级分类

数据中心的整体设计通常被归类为“Tier I-IV” [67]。

- Tier I数据中心的电力和冷却分配只有一条路径，没有冗余组件。
- Tier II在这个设计上增加了冗余组件（N +1），提高了可用性。
- Tier III 数据中心具有多个电力和冷却分配路径，但只有一条活动路径。 它们还具有冗余组件，并且在维护期间可以同时维护，通常采用 N +2 设置提供冗余。
- Tier IV 数据中心具有两条活动的电力和冷却分配路径，在每条路径上具有冗余组件，并且应该能够容忍任何单个设备故障而不影响负载。

这些层级分类并不是100%准确的。 大多数商业数据中心在层级 III 和 IV 之间选择平衡建设成本和可靠性。 现实世界中的数据中心可靠性也受到运营组织的质量的强烈影响，而不仅仅是数据中心的设计。 行业中使用的典型可用性估计范围从 II 级数据中心的 99.7% 可用性到 III 级和 IV 级的 99.98% 和 99.995%。

数据中心的规模差异很大。 美国的服务器中有三分之二存放在小于5,000平方英尺（450平方米）且关键功率小于1兆瓦的数据中心中[26]（第27页）。 大多数大型数据中心都是为多家公司的服务器提供托管服务（通常称为共享数据中心或“colos”），并且可以支持10-20兆瓦的关键负载。如今，很少有数据中心超过30兆瓦的关键容量。

4.2 数据中心电力系统

图4.1显示了典型数据中心的组成部分。 电力从外部变压器进入建筑物，通常位于公用事业的变电站；这部分电力系统通常被称为“中压”（通常为10-20千伏），以区别于高压长距离输电线路（60-400千伏）和“低压”内部电力分配（110-600伏）。中压线路终止于主开关设备，其中包括断路器以防止电气故障和变压器将电压降低到400-600伏。然后，低压电力流入不间断电源（UPS）系统，该系统还从一组柴油发电机获得第二个供电（相同电压），当公用事业电力中断时，这些发电机将开始运行。

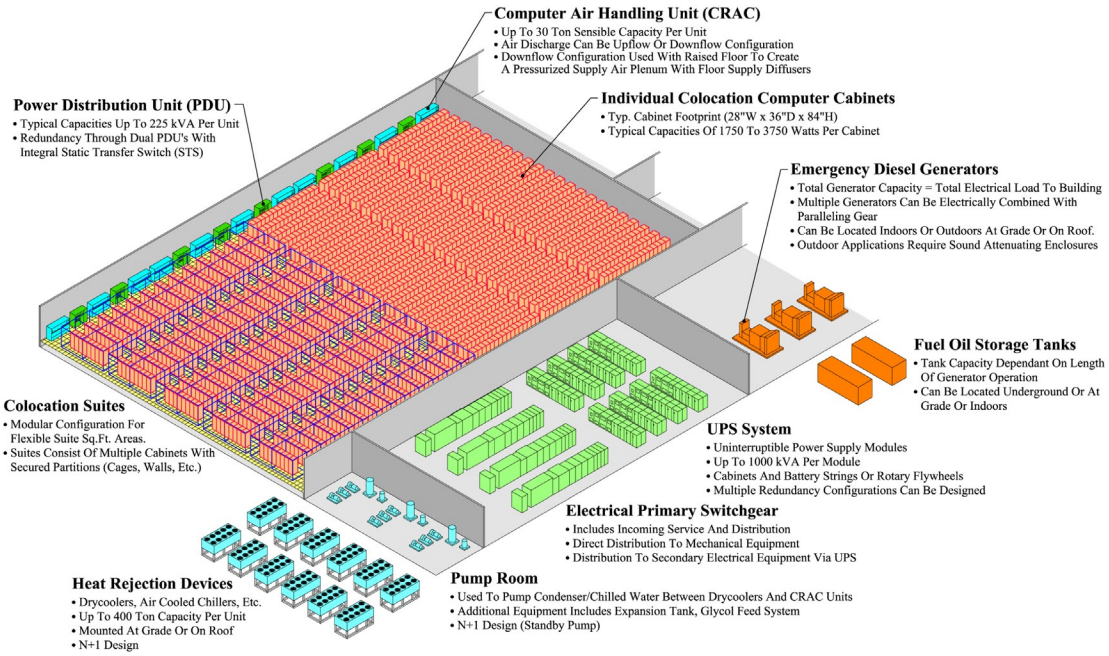

图**4.1**：典型数据中心的主要组件（图片由DLB Associates [23]提供）

4.2.1 UPS系统

UPS通常将三个功能合并在一个系统中

- 首先，它包含一个转换开关，选择活动电源输入（可以是公用事业电源或发电机电源） 在停电后，转换开关会感知到发电机已经启动并准备提供电源；通常，发电机需要10-15秒启动并承担额定负载
- 其次，UPS包含电池或飞轮，以弥补公用事业故障和发电机电源可用之间的时间 典型的UPS通过AC-DC-AC双转换来实现这一点；即，输入的交流电被转换为直流电，然后供电给UPS内部的直流母线，该母线还连接到一系列电池 然后，直流母线的输出被转换回交流电以供应数据中心设备 因此，当公用事业电源故障时，UPS失去输入（交流）电源，但保留内部直流电源，因为电池仍然提供电源，并在第二个转换步骤后保留交流输出电源最终，发电机启动并重新供应输入的交流电源，减轻了数据中心负载的UPS电池的压力。
- 第三，UPS对输入电源进行调节，消除交流电源中的电压峰值、电压下降或谐波失真。 这种调节通常通过双重转换步骤来完成。

由于UPS电池占用了相当大的空间，UPS通常安置在与数据中心楼层分开的UPS机房中。典型的UPS容量范围从几百千瓦到2兆瓦。

4.2.2 电源分配单元

然后，UPS输出被路由到放置在数据中心楼层上的电力分配单元（PDUs）。PDUs类似于住宅中的断路器面板：它们接收高电压供电（通常为200-480V），并将其分成许多110V或220V的电路，供应实际的服务器。 每个电路都由自己的断路器保护，这样服务器或电源的接地短路只会触发该电路的断路器，而不会触发整个PDU甚至UPS的断路器。典型的PDU负载为75-225千瓦，而典型的电路负载为20或30安培，110-220V，即最大6千瓦。PDUs通常通过接受两个独立的电源（通常称为“A面”和“B面”）并能够在它们之间进行非常小的延迟切换来提供额外的冗余，以便一侧电源的故障不会中断服务器的电源。在这种情况下，数据中心的UPS单元被复制成A面和B面，即使UPS发生故障，也不会中断服务器的电源。

现实世界的数据中心包含许多简化设计的变体，如下所述。 典型的变体包括发电机或UPS单元的“并联”安排，即多个设备共享一个总线，以便其他设备可以接管故障设备的负载，类似于RAID系统中处理磁盘故障的方式。 常见的并联配置包括N +1配置（允许一个故障或维护），N +2配置（即使一个单元离线维护，也允许一个故障），以及2N（冗余对）。

4.3 数据中心的冷却系统

冷却系统比电力系统要简单一些。 通常，数据中心的地板会被抬高，即在混凝土地板上安装一个靠立柱支撑的钢格栅（如图4.2所示）。地板下的区域通常用于将电力电缆引导到机架，但其主要用途是向服务器机架分发冷气。

4.3.1 CRAC单元

CRAC单元（CRAC是20世纪60年代的一个术语，用于计算机机房空调）通过向空气流道中吹送冷气来增压提升地板。 这种冷气通过放置在服务器机架前面的穿孔瓷砖从空气流道中逸出，然后流经服务器，服务器在后方排出温暖的空气。 机架排列成长走廊，冷走廊和热走廊交替排列，以避免冷热空气混合。（将冷气与热气混合会降低冷却效率；一些较新的数据中心甚至用墙壁隔离热走廊，以避免热气泄漏。

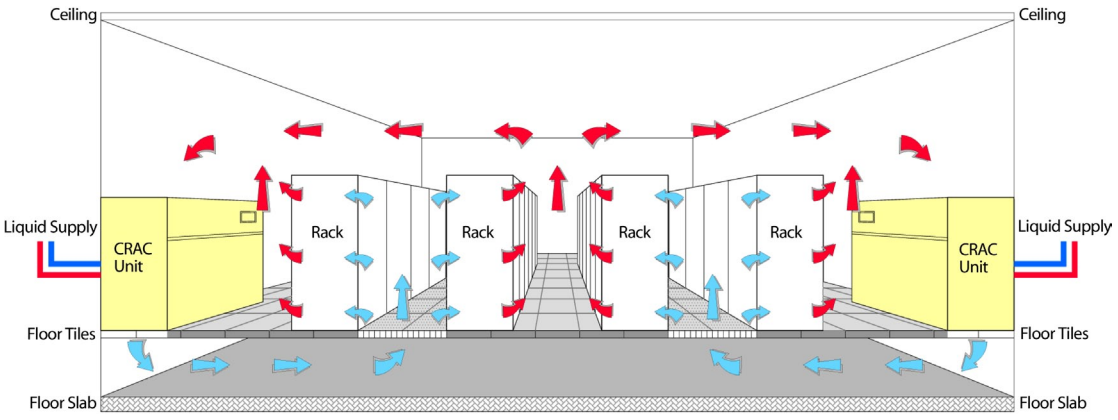

图4.2：带有冷热走廊设置的数据中心提升地板（图片由DLB Associates [23]提供）。

最终，服务器产生的热空气重新循环回CRAC单元的进气口，被冷却后再次排出冷气到提升地板的空气流道中。

CRAC单元由液冷剂泵送的线圈组成；风扇通过这些线圈推动空气，从而冷却它。一组冗余泵将冷冷却剂循环到CRAC，并将温暖的冷却剂送回冷却器或冷却塔，将热量排出到室外环境。
通常，进入的冷却剂温度为12-14°C，空气从CRAC中排出的温度为16-20°C，导致冷走道（服务器进气）的温度为18-22°C。（服务器进气温度通常高于CRAC出口温度，因为在到达服务器之前它会稍微加热，并且由于无法消除的再循环。）温暖的冷却剂返回到冷却器再次冷却到12-14°C。

4.3.2 自由冷却

较新的数据中心通常在冷凝器水回路中插入冷却塔，使用“自由冷却”预冷冷却剂，然后再到达冷却器。 自由冷却并不是真正免费的，但它比使用冷却器更节能。

基于水的自由冷却使用冷却塔来散热。 冷却塔使用一个独立的冷却回路，在其中水在热交换器中吸收冷却剂的热量。 在冷却塔中，温水流经一个大面积结构，并通过蒸发将热量传递给外部空气，从而冷却下来。 请注意，如果空气相对干燥，温水可以冷却到低于环境温度的温度。当空气经过水流时，蒸发会降低水的温度接近“湿球”空气温度，即低于环境“干球”空气温度。 冷却塔在温和的气候和低湿度的气候中效果最好；具有讽刺意味的是，在非常寒冷的气候中，它们的效果不如预期，因为需要额外的机制来防止冷却塔上结冰。

另外，自由冷却系统可以使用基于乙二醇的散热器安装在建筑物外部来散热。 这在寒冷的气候中非常有效（比如芝加哥的冬天），但在温和或温暖的温度下效果较差，因为通过空气对空气的对流换热比蒸发换热效果差。 另外，一些设计跳过热交换步骤，通过CRAC直接使用大型风扇将外部空气推入房间或提升地板空间，当外部温度允许时（有关这一领域的极端实验，请参见[2]）。

大多数机械冷却设备也备有发电机的支持（有时还有UPS装置），因为数据中心在超过几分钟没有冷却之前无法正常运行。 在典型的数据中心中，冷却器和泵可以增加40%或更多需要由发电机支持的关键负载。

4.3.3 空气流量考虑

大多数数据中心使用上升地板设置如上所述。 要更改传递给特定机架或行的冷却量，我们只需通过用通气瓷砖替换实心瓷砖或反之亦然来调整一行中的通气瓷砖数量。 为了使冷却工作良好，通过瓷砖进入的冷气流需要与机架内的服务器的水平气流相匹配——例如，如果一个机架有10台每台气流为100立方英尺/分钟的服务器，则通气瓷砖的净流出量应为1,000立方英尺/分钟（如果通向服务器的空气路径没有得到严格控制，则应更高）。 如果净流出量较低，所有冷气将被机架底部的一些服务器吸入，而顶部的服务器将吸入来自机架上方的热气；这种不良效应通常被称为“循环”，因为热气从一个服务器的排气口循环到相邻服务器的进气口。 一些数据中心通过位于服务器机架上方的管道提供额外的冷气来解决这个问题。

这种对匹配空气流动的需求限制了数据中心的功率密度。 对于服务器之间的固定温差，机架的空气流量需求随着机架功耗的增加而增加，并且通过提升地板瓷砖提供的空气流量必须与功耗成线性关系增加。 这反过来增加了我们需要在地板下腔室中产生的静压力的数量，从而增加了CRAC（计算机房空调）所需的风扇功率来将冷气推入腔室。 在低密度下，这很容易实现，但在某一点上，物理定律开始限制我们，使进一步增加压力和空气流量在经济上变得不切实际。 通常情况下，这些限制使得很难在不大幅增加成本的情况下超过每平方英尺150-200瓦的功率密度。

如前所述，新型数据中心已经开始将热通道与机房物理隔离，以消除空气再循环并优化返回CRAC的路径。在这种设置中，整个机房都充满了冷气（因为温暖的排气被保留在单独的腔室或管道系统内），因此，机架中的所有服务器都以相同的温度吸入空气[63]。

4.3.4 机架内冷却

机架内冷却产品是将整个房间充满冷气的一种变体，还可以提高功率密度和冷却效率，超越传统的提升地板限制。 通常情况下，机架内冷却器在机架的后部添加了一个空气-水热交换器，使热空气从服务器排出后立即流过被水冷却的线圈，从而实质上绕过了服务器排气和CRAC输入之间的路径。 在某些解决方案中，这种额外的冷却只去除部分热量，从而降低了房间CRAC的负荷（即降低了CRAC所感知到的功率密度），而在其他解决方案中，它完全去除了所有热量，实际上取代了CRAC。这些方法的主要缺点是它们都需要将冷却水引入每个机架，大大增加了管道成本，并增加了在数据中心地板上放置可能泄漏的耦合件的担忧。

4.3.5 基于容器的数据中心

基于容器的数据中心通过将服务器机架放入标准集装箱，并将热交换和电力分配集成到集装箱中，进一步超越了机架内冷却。 与完全机架内冷却类似，集装箱需要供应冷却水，并使用线圈从流过其上的空气中移除所有热量。空气处理类似于机架内冷却，并且通常允许比常规提升地板数据中心更高的功率密度。 因此，基于容器的数据中心在一个小包装中提供了典型数据中心房间的所有功能（机架、CRAC、PDU、布线、照明）。 与常规数据中心房间一样，它们必须通过外部基础设施（如冷却器、发电机和UPS装置）来完全发挥功能。最近曝光的消息显示，谷歌已经建造了一个基于容器的数据中心，自2005年以来一直在运营[33]，尽管这个想法可以追溯到谷歌在2003年的专利申请。

与今天的典型数据中心相比，[33]中描述的基于容器的设施在能源效率方面取得了极高的评级，我们将在下一章中进一步讨论。 微软还宣布将在新的数据中心中大量使用容器[92]。

第5章

能源和功率效率

能源效率长期以来一直是移动和嵌入式领域的主要技术驱动因素，但对于通用计算来说，这是一个相对较新的关注点。 早期的工作强调延长电池寿命，但后来扩展到包括减少峰值功率，因为热约束开始限制进一步的CPU性能改进。 能源管理现在是服务器和数据中心运营的一个关键问题，重点是减少所有与能源相关的成本，包括资本、运营费用和环境影响。 为移动设备开发的许多节能技术是解决这个新问题领域的自然选择，但最终，仓库规模的计算机（WSC）与移动设备有很大的区别。 在本章中，我们描述了数据中心级别到组件级别的WSC能源和功率效率的一些最相关方面。

5.1 数据中心能效

WSC的能效广义上定义为在过程中所执行的计算工作量与总能量消耗之比。 这就是绿色网格数据中心性能效率（DCP*E*）试图捕捉的概念[37,38]。 尽管尚未定义实际的度量标准，但一般的想法是运行一个标准化的数据中心工作负载（例如SP*E*C或TPC基准测试），并测量总功耗。 虽然这是一个有用的观点，但我们对DCP*E*作为一个实际度量标准是否会有很大影响持怀疑态度，因为很难进行测量 - 很少有机构能够测量它，因为运行在它们数据中心中的服务器正在运行实际应用程序，因此无法进行基准测试。

相反，我们发现将DCPE分解为三个可以由适当的工程学科独立测量和优化的组成部分更有用，如下方的方程所示：

$$\text{效率} = \frac{\text{计算}}{\text{总能量}} = \underbrace{\left(\frac{1}{\text{PUE}}\right)}_{\text{(a)}} \cdot \underbrace{\left(\frac{1}{\text{SPUE}}\right)}_{\text{(b)}} \cdot \underbrace{\left(\frac{\text{计算}}{\text{电子组件的总能量}}\right)}_{\text{(c)}}$$

方程式**5.1**：将能效指标分解为三个组成部分：设施项(a)，服务器能量转换项(b)，以及电子组件在执行计算本身时的效率(c)。

效率计算中的第一个项(a)是功耗使用效率(PUE)，它反映了数据中心建筑基础设施的质量[37]，并捕捉了总建筑功耗与IT功耗的比值，即实际计算设备（服务器、网络设备等）消耗的功耗。文献中也将IT功耗称为关键功耗。

由于PUE因素对正在执行的计算的性质视而不见，它们可以通过电力监测设备进行客观和持续的测量，而不会对正常运行造成任何干扰。 可悲的是，平均数据中心的PUE非常糟糕；根据一项2006年的研究[51]，估计目前85%的数据中心的PUE大于3.0，也就是说，建筑的机械和电气系统消耗的电力是实际计算负载的两倍；只有5%的数据中心的PUE为2.0。其他研究显示了稍微好一些的情况，对22个数据中心进行的调查显示平均PUE约为2.0（Greenberg等人[41]和Tschudi等人[86]）。 图5.1显示了Greenberg最近调查（2007年）的PUE值。

高PUE值是由多个开销来源造成的（见图5.2）。在典型的架空地板数据中心中，冷却器消耗了最大比例的功耗开销，通常为IT负载的30-50%。 机房空调（CRAC）单元紧随其后，消耗IT负载的10-30%的功耗。

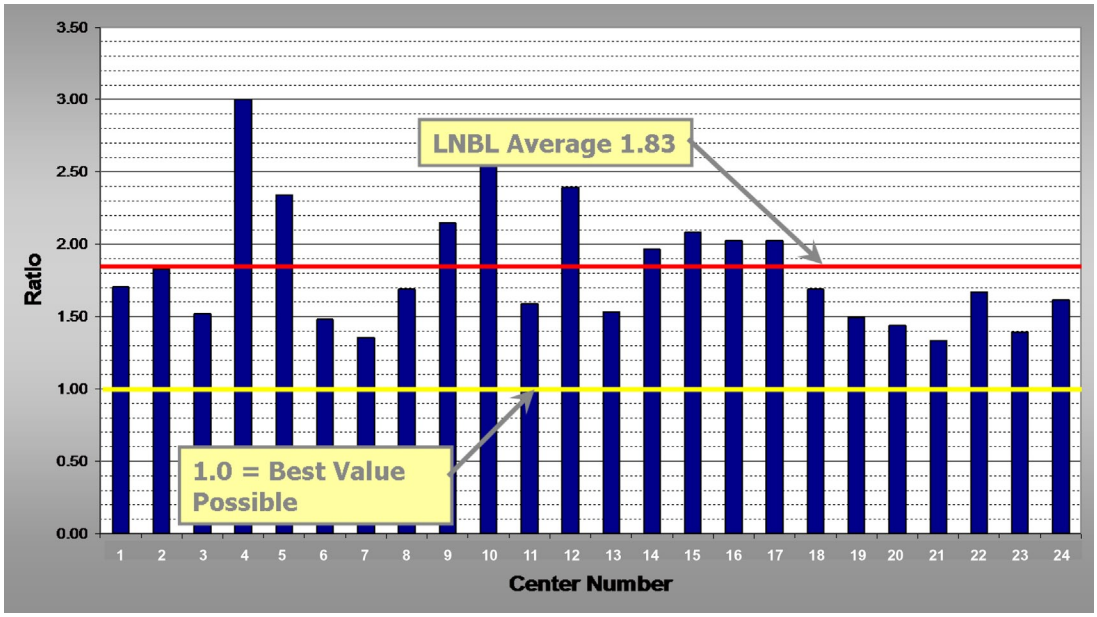

图5.1：2007年24个数据中心的功耗效率调查（Greenberg等人）[41]。

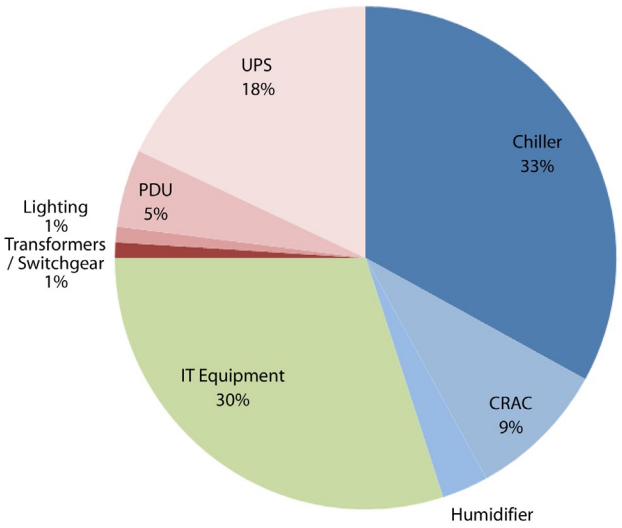

图5.2：数据中心能源开销的细分（ASHRAE）。

负载（主要是风扇），其次是UPS系统，通过交流-直流-交流转换损耗消耗7-12%的关键功率（当UPS轻载时，相对损耗更高）。
其他设施元素[加湿器、电源分配单元（PDUs）、照明]进一步提高了PUE水平。 这种低效率主要是由于历史上对效率的忽视，而不是由物理上的固有限制所造成的。 人们普遍认为，一个设计良好、运营良好的数据中心的PUE应该低于2，而2007年美国环保署关于数据中心功耗的报告指出，在“先进技术”场景下，到2011年可以实现PUE为1.4 [26]。 最明显的改进机会是使用蒸发冷却塔、更高效的空气流动以及消除不必要的功率转换损耗。

5.1.1 数据中心效率损失的来源

为了说明问题，让我们来看一下典型数据中心的效率损失来源[41]。 将输入的高压电从115 kV降压到中压配电线路（通常在美国为13.2 kV）的变压器非常高效，将其进一步降压到480 V的变压器也是如此。在这两种情况下，变压损耗通常低于半个百分点。 不间断电源（UPS）是大多数转换损耗的来源，在最佳情况下的效率为88-94%（如果负载较轻，则效率更低）。旋转式

UPS装置（飞轮）和高效率UPS装置在正常运行时通过绕过UPS可以达到约97%的效率。最后，如果相应的电缆很长，将低压电（110或220 V）引入机架时可能会损失相当多的电力。

回想一下，一个大型设施可以有一个地板面积超过100米长或宽，并且根据距离和电缆类型，这些电缆可能会损失1-3%的功率。

然而，大部分的效率损失都出现在冷却方面。 将冷气从CRACs输送到设备架上的长距离消耗了大量的风扇功率，将温暖的空气送回CRAC进气口也是如此。 更糟糕的是，在这些长路径中，冷气和温暖的空气可能会混合在一起，大大降低CRAC装置的效率，减少CRACS的温度差。 同样，将数据中心保持在非常凉爽的常规做法需要接近10°C的冷却水温度，增加了冷却机的负荷。 这样低的温度也会导致CRAC装置的线圈上出现冷凝，进一步降低其效果，并且具有讽刺意味的是，需要在其他地方花费额外的能源进行重新加湿。

5.1.2 提高数据中心能源效率

精心设计以提高效率可以大幅改善PUE [57,66,39]。 尽管大多数数据中心的PUE值为2或更高，但仍有可能建造更高效的数据中心。 提高冷通道温度至25-27°C而不是传统的20°C是改善效率最简单的步骤之一。实际上，几乎没有服务器或网络设备需要20°C的进气温度，而且对于这些温度范围，更高温度导致设备故障的担忧几乎没有实证证据支持。 提高冷通道温度可以提高冷却水温度，改善冷却机的效率，并在与自由冷却结合时减少其运行时间。 同样，有效管理热排出热量可以极大地提高冷却效率；这是容器化数据中心更高效的主要原因之一。 最后，通过选择更高效的设备，通常可以大大减少UPS和电力分配的损耗。最近几个月，有几家公司宣布其数据中心的PUE值低于1.3 [57]，而谷歌发布的数据显示谷歌设计的数据中心的年平均PUE为1.2 [32]。

2009年4月，谷歌公布了其数据中心架构的细节，包括一个基于容器的数据中心的视频导览[33]。 这个数据中心在2008年实现了1.24的最先进的年PUE，但与传统数据中心只有几个主要方面的不同：

- 精心处理空气流动：服务器排出的热空气不允许与冷空气混合，通往冷却线圈的路径非常短，因此几乎不需要花费能量来移动冷空气或热空气的长距离。

- 提高冷走廊温度：容器的冷走廊保持在大约27°C，而不是18-20°C。 更高的温度使得数据中心更容易高效冷却。
- 使用自由冷却：几个冷却塔通过蒸发水来散热，大大减少了运行冷却机的需求。 在大多数温和气候中，冷却塔可以消除大部分冷却机的运行时间。 谷歌在比利时的数据中心甚至完全取消了冷却机，100%的时间都使用“免费”冷却。

- 每个服务器都包含一个小型UPS，即一个浮动在服务器电源的直流侧的电池，效率达到99.99%。 这些每台服务器的UPS消除了整个设施UPS的需求，将整体电源基础设施的效率从约90%提高到接近99% [34]。

起初，这个数据中心可能与传统的数据中心完全不同，从某种意义上说是真的，但几乎所有的技术都可以应用于更传统的设计（例如，没有集装箱，没有每台服务器的UPS）。即使在一个看起来很传统的数据中心中，这些技术也应该能够实现PUE在1.35到1.45之间，即远高于当前行业平均水平的效率。

尽管PUE可以捕捉到设施的开销，但它并不考虑IT设备本身的低效率。 服务器和其他计算设备实际上并没有使用其输入功率的100%进行计算。 特别是在服务器的电源、电压调节模块（VRM）和散热风扇中可能会损失大量功率。 效率计算中的第二项（b）考虑了这些开销，使用类似于PUE的度量标准，但应用于计算设备：服务器PUE（SPUE）。它由总服务器输入功率与其有用功率的比值组成，其中有用功率仅包括与计算直接相关的电子组件消耗的功率：主板、硬盘、CPU、DRAM、I/O卡等。换句话说，有用功率不包括电源、VRM和风扇的所有损耗。 目前还没有常用的SPUE测量协议，尽管Climate Savers Computing Initiative（climatesaverscomputing.org）正在研究一种。今天的服务器的SPUE比率通常为1.6-1.8；许多电源的效率低于80%，许多主板使用的VRM同样低效，电能转换损失超过30%。 相比之下，最先进的SPUE应该低于1.2 [17]。

因此，PUE和SPUE的组合构成了设施和计算设备的总电机械开销的准确评估。 这种真实的（或总）PUE指标（TPUE），定义为PUE ∗SPUE，目前平均每个数据中心为3.2以上；也就是说，每个有生产力的瓦特至少消耗另外2.2瓦特！相比之下，一个PUE为1.2且SPUE为1.2的设施将使用不到一半的能源。但这仍然不是理想的

因为只有70%的能源用于实际计算，所以这还不是理想的，但相比现状已经有了很大的改进。根据当前技术水平，每年的TPUE可能代表了在实际环境中经济可行的上限，大约为1.25。

5.2 计算效率的测量

到目前为止，我们主要讨论了电方面的效率，基本上忽略了方程5.1的项(c)，该项衡量了系统中电子组件所接收到的电能有多少被转化为有用的工作。 这最后一项可以说是最难以客观测量的，因为计算系统具有通用性质。 最终，我们希望衡量在计算中所消耗的能量所获得的价值量。 这样的测量可以用于比较两个仓库规模计算机的相对效率，或者指导新系统的设计选择。

不幸的是，没有两家公司运行相同的工作负载，现实世界的应用程序组合也在不断变化，因此如果目标是比较两个仓库规模计算机，使用现实世界的数据进行基准测试是很困难的。在高性能计算（HPC）领域，最近有一个尝试开始使用现有的HPC基准测试（LINPACK）来对全球顶级超级计算机的能源效率进行排名，这被称为Green 500 [36]。 我们不知道是否有类似的倡议适用于互联网服务。

缺乏有意义的集群级基准测试并不妨碍获得有意义的能效测量。 现有的设施可以通过负载测试其服务并充分仪器化设施以测量或估计能源使用情况来测量其自身的能效。 在缺乏标准化的集群级基准测试的情况下，服务器级基准测试也可以有用，只要能进行有意义的推断。 最近发布的两个基准测试为服务器的能效核算提供了一个良好的起点：Joulesort [80] 和 SPECpower_ssj2008 基准测试 [79]。 Joulesort 包括测量执行一个离线排序所需的总系统能量，并试图推导出一个指标，使得可以比较从嵌入式设备到超级计算机的系统。SPECpower 相反，专注于服务器级系统，并计算在企业级 Java 平台上运行典型业务应用程序的性能与功耗比。 类似的能效基准测试需要为其他重要的计算基础设施部件（如网络交换机和存储子系统）进行开发。 存储网络行业协会正在努力开发类似的网络存储产品基准测试 [40]。

5.2.1 一些有用的基准测试

显然，同一个应用程序二进制文件在不同的服务器架构上可能消耗不同的功率，同样，一个应用程序在服务器的容量方面可能消耗更多或更少，取决于软件性能调优。 像SPECpower_ssj2008这样的基准测试提供了一个代表广泛的服务器工作负载类别的标准应用程序基础，它可以帮助

我们隔离硬件平台的效率差异。 特别是，SPEC功耗报告规则有助于突出当前服务器的一个关键能源使用特性：在低利用率下，计算系统的效率明显低于在最大利用率下运行时的效率。 与大多数性能基准不同，SPEC功耗要求不仅在峰值利用率下报告每瓦性能，而且在整个利用率范围内报告（以10%的间隔）。

图5.3显示了截至2008年6月的最佳条目的SPEC功耗基准测试结果。 结果显示了两个指标：性能（每秒事务数）与功耗比和平均系统功耗，在11个负载水平上绘制。 图中一个值得注意且与其他SPEC功耗基准测试结果相同的特点是：随着目标负载的降低，性能与功耗比急剧下降，因为系统功耗下降的速度比性能下降的速度慢得多。

请注意，例如，在30%负载下的能源效率不到100%效率的一半。
此外，当系统处于空闲状态时，它仍然消耗接近175瓦，这超过了服务器峰值功耗的一半！

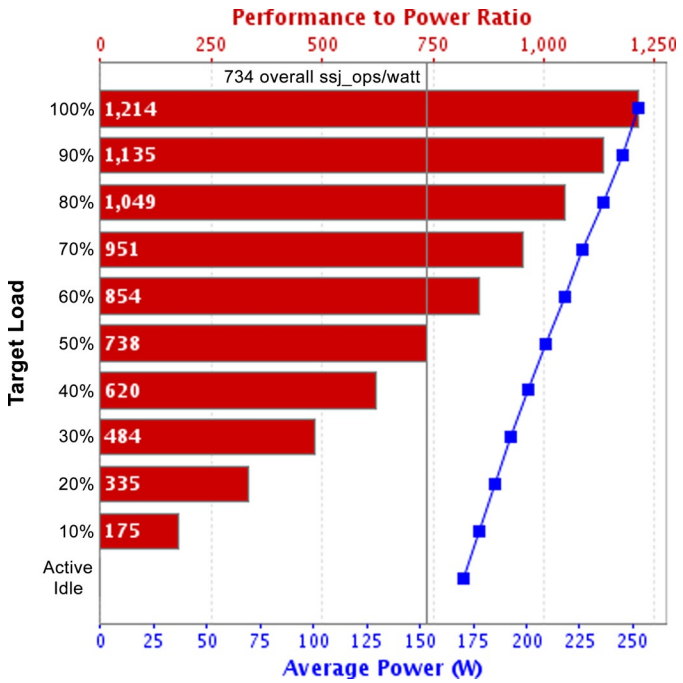

图5.3：SPECpower_ssj2008的一个示例基准测试结果；柱状图表示能源效率，而线条表示功耗。 两者都绘制了一系列利用率水平，平均指标对应垂直的深色线。 该系统配备了一颗单芯片2.83 GHz四核Intel Xeon处理器，4 GB DRAM和一个7.2 k RPM 3.5″SATA硬盘驱动器。

5.2.2 负载 vs. 效率

图5.4显示了SPEC power基准测试中前10个条目在30%负载下与100%负载下的相对效率，以及空闲功耗与峰值功耗的比率-最右边的值是10个结果的平均值。 图5.3中的系统行为对现代服务器类机器来说很常见，与我们在Google服务器中观察到的情况类似。 如果服务器平均运行非常接近峰值负载水平，这种行为是可以接受的。

不幸的是，大多数数据中心运营商报告称情况并非如此。

图5.5可以被视为WSC的一个示例负载配置文件。它显示了在6个月期间5,000台Google服务器的平均CPU利用率。 尽管曲线的形状在不同的集群和工作负载之间有所变化，但一个共同的趋势是，平均而言，服务器在高负载水平上花费的总时间相对较少。 相反，大部分时间都在10-50%的CPU利用率范围内。 这种活动配置文件与现代服务器的能效配置文件完全不匹配，因为它们大部分时间都在它们最低效的负载区域内。

在WSC的能源使用情况中还有另一个特点，这在图5.4中并不明显；这些系统中的单个服务器也很少完全闲置。 例如，考虑一个大型的网络搜索工作负载，就像第2章中描述的那样，查询被发送到非常多的服务器，每个服务器在整个索引中搜索其本地切片。

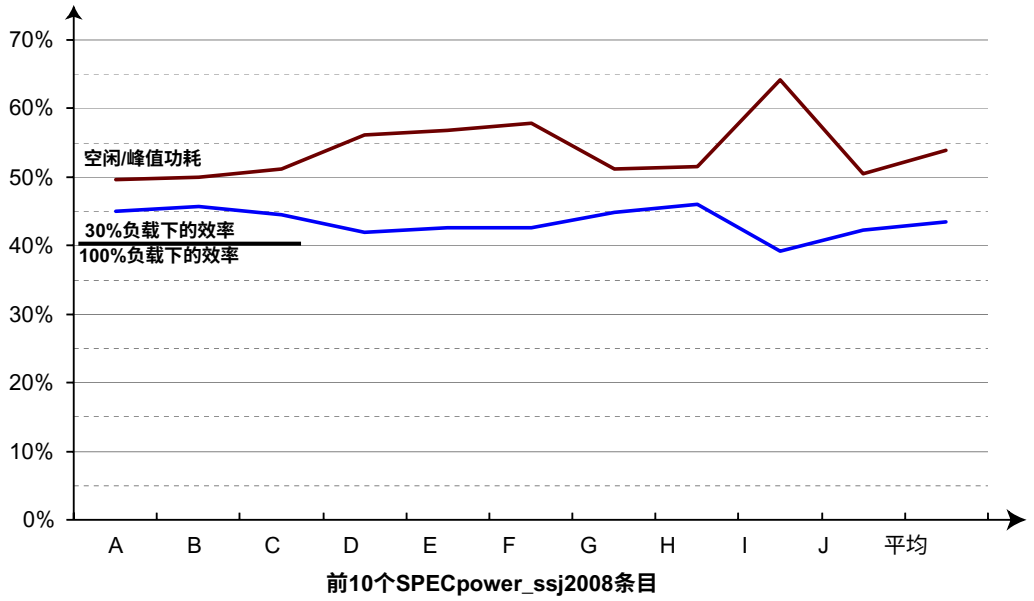

图5.4：30%负载下的空闲/峰值功耗和能效（相对于100%负载效率）前10个SPECpower_ssj2008条目的数据（来自2008年中期）

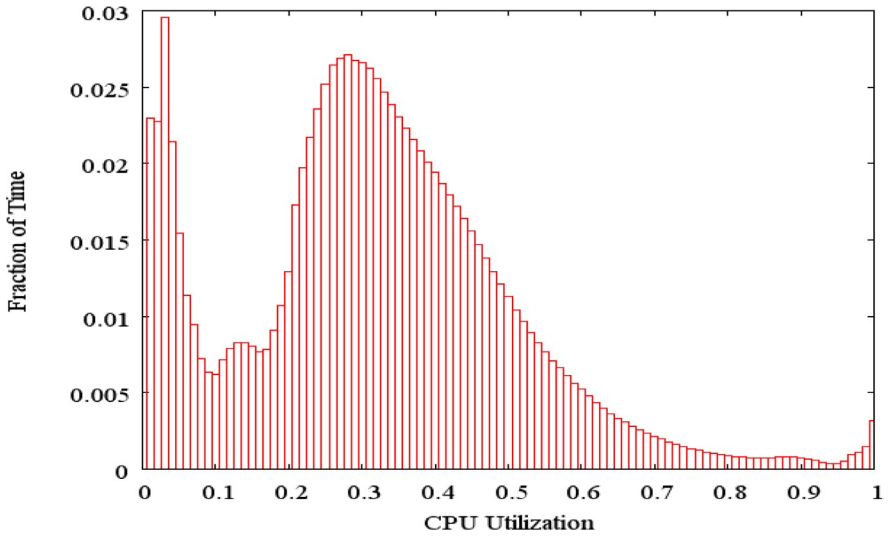

图5.5：在6个月的时间内，对5000台Google服务器的活动概况。

dex. 当搜索流量很大时，所有服务器都被大量使用，但在低流量期间，一个服务器可能仍然每秒收到数百个查询，这意味着任何空闲时间都不会超过几毫秒。

尽管存在低活动期间，但显著的空闲间隔的缺失主要是由于将高性能、稳健的分布式系统软件应用于合理的设计原则。 大规模的互联网服务依赖于对大量服务器的高效负载分配，这种情况下，当负载较轻时，我们往往会在多个服务器上分散负载，而不是集中负载在少数服务器上并使其闲置。 在低活动期间，应用程序（或底层的集群管理系统）可以通过将工作负载及其相应的状态迁移到较少的机器上来制造空闲。 当使用简单的复制模型时，这可以相对容易实现，当服务器主要是无状态的（即，提供存储在共享NAS或SAN存储系统上的数据）。 然而，对于更复杂的数据分布模型或对数据局部性进行积极利用的模型，这将以软件复杂性和能源成本为代价。

在大规模分布式系统中，难以制造有用的空闲期的另一个原因是需要具有弹性的分布式存储。 Google文件系统（GFS [31]）通过将给定文件的数据块副本分布在整个集群中而不是仅集中在少数几台机器上，实现了更高的弹性。 这有利于文件系统的性能，因为它实现了细粒度的负载平衡，以及弹性，因为当存储服务器出现故障时

当系统崩溃（或磁盘故障）时，该系统中的副本可以由数千台机器重建，使恢复过程非常高效。 这种本来合理的设计的后果是，低流量水平会导致所有机器的活动减少，而不是其中一个重要子集的完全闲置。 还有一些实际考虑因素可能会阻碍完全闲置，因为网络服务器经常在定期间隔上执行许多小的后台任务。 关于无滴答内核项目的报告[78]提供了其他难以创建和维护闲置的示例。

5.3 能效比例计算

在早期的一篇文章中 [6]，我们认为服务器工作负载和服务器能效行为之间的不匹配主要需要在硬件层面上解决；仅靠软件无法高效利用仅在非活动空闲模式（睡眠或待机）或全速运行时才高效的硬件系统。 我们认为系统在轻负荷使用时效率低下，主要是因为工程师和研究人员对该领域的能效重要性缺乏认识。

我们建议将能效比例作为计算组件的设计目标之一。 理想情况下，能效比例系统在空闲时几乎不消耗电力（特别是在仍然可用于工作的活动空闲状态下），并且随着活动水平的增加逐渐消耗更多电力。 对于这个理想曲线的简单推理是假设活动和功耗之间存在线性关系，没有常数因素。 这种线性关系将使能效在活动范围内保持均匀，而不会随着活动水平的降低而下降。 然而，请注意，线性关系不一定是节能的最佳关系。

从图5.5可以看出，可以认为由于服务器在高活动水平上花费的时间相对较少，即使效率随着活动水平的增加而降低，尤其是在接近最大利用率时，这可能也是可以接受的。

图5.6展示了两个假设系统的可能能源效率，这些系统比典型服务器更具能源比例。 红色曲线对应于典型服务器，例如图5.3中的服务器。绿色曲线显示了更具能源比例系统的归一化功率使用和能源效率，该系统仅在峰值功率的10%处空闲，并具有线性功率与负载行为。 请注意，其效率曲线比典型服务器的曲线要好得多；尽管其效率仍随负载水平下降，但下降速度要缓慢得多，并且在峰值负载的30%处保持相对较高的效率水平。 蓝色曲线显示的系统也在峰值的10%处空闲，但在负载水平为0%至50%之间的负载水平区域具有次线性功率与负载关系。 该系统的效率曲线峰值不在100%负载处，而是在30-40%的负载区域。 从能源使用的角度来看，这种行为与图5.5中所示的WSC活动谱的类型是相匹配的。

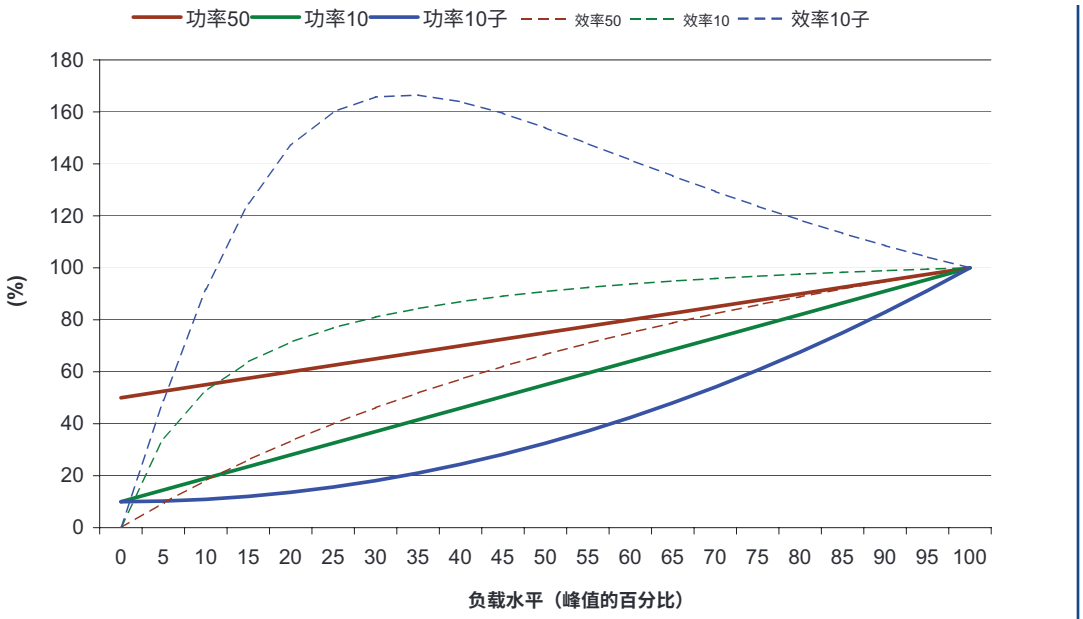

图5.6：三个假设系统的功率及相应功率效率：典型服务器的空闲功率为峰值的50%（Pwr50和Eff50），更具能源比例的服务器的空闲功率为峰值的10%（Pwr10和Eff10），以及具有亚线性能源比例的服务器的空闲功率为峰值的10%。

Fan等人在他们的功率配置研究中评估了WSCs中能源比例的潜在收益。 他们使用数千台机器在6个月内的活动水平跟踪来模拟使用更具能源比例的服务器所获得的节能效果-空闲功耗为峰值的10%（类似于图5.6中的绿色曲线），而不是50%（如相应的红色曲线）。 他们的模型表明，仅通过增加能源比例，能源使用量将减少一半，因为两台比较的服务器具有相同的峰值能源效率。

5.3.1 能效比机器的动态功率范围

能效比机器将展示出一个宽广的动态功率范围——这在计算设备中是罕见的特性，但在其他领域并不是没有先例。 例如，人类的平均每日能量消耗接近于一台旧个人电脑：大约120瓦。然而，处于休息状态的人类可以消耗低至70瓦，同时能够持续数十分钟的高峰功率超过1千瓦，据报道，顶级运动员甚至可以达到2千瓦[54]。图5.7列出了

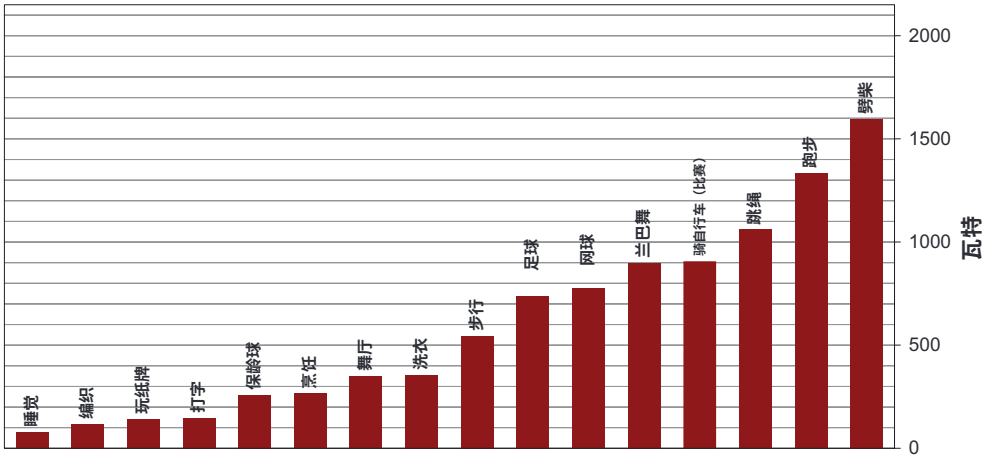

图5.7：人类能量使用与活动水平（成年男性）[54]。

几种职业活动及其对应的能量消耗，说明成年男性的动态功率范围几乎是今天典型计算机的2倍×，相比之下，今天典型计算机的功率范围只有2倍×。 普遍认为，能源效率是进化过程中受欢迎的特性之一；我们想知道能源比例是否可能是实现更高能源效率的一种成功手段。

5.3.2 低能源比例的原因

尽管CPU在能源使用方面历史上声名狼藉，但它们不一定是能源比例不佳的主要原因。 例如，早期的谷歌服务器将总能源预算的60%用于CPU芯片，而如今它们往往使用不到50%。

在过去几年中，CPU设计师对能源效率的关注超过了其他子系统的设计师。 与继续追求更高的时钟频率和更大的推测执行级别相比，转向多核架构是这种更节能趋势的原因之一。

图5.8显示了最近一台Google服务器的主要子系统在计算负载从空闲到满负荷时的功耗使用情况。 当处于峰值时，CPU对系统功耗的贡献接近50%，但在低活动水平下降到不到30%，使其成为所有主要子系统中最节能的部分。 根据我们的经验，服务器级别的CPU的动态功耗范围通常大于3.0×（在这种情况下超过3.5×），而面向嵌入式或移动市场的CPU则可以做得更好。

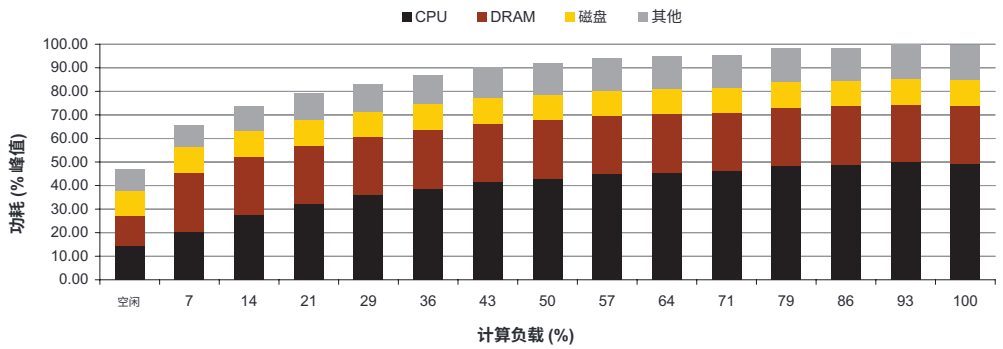

图5.8：当计算负载从空闲变化到满负载时，在×86服务器中的子系统功耗使用情况。

相比之下，内存系统、磁盘驱动器和网络设备的动态范围要低得多：内存约为2.0×，磁盘约为1.3×，网络交换机则低于1.2×。 这表明，仅通过CPU优化无法实现系统级的能源比例性，而是需要在所有组件上进行改进。 这表明，仅通过CPU优化无法实现系统级的能源比例性，而是需要在所有组件上进行改进。

5.3.3 如何改善能源比例

将能源比例作为一个优点的关注点可能通过渐进的工程改进在整个系统组件上带来改进。然而，在某些情况下，可能需要更大的创新。 例如，磁盘驱动器在保持盘片旋转方面花费了大部分能源预算，可能高达总功率的70%（对于高转速驱动器而言）。 创建额外的能源效率和能源比例可能需要更小的旋转速度、更小的盘片或使用多个独立的磁头组件的设计。

Carrera等人[11]考虑了多速度驱动器和多种服务器和笔记本驱动器组合以实现比例能源行为的能源影响。 最近，Sankar等人[81]探索了磁盘驱动器的不同架构，观察到由于磁头移动相对能源成比例，具有较低旋转速度和多个磁头的磁盘与单磁头高转速磁盘相比，可以实现类似的性能和更低的功耗。最后，我们应该提醒读者，能源比例行为不仅是电子组件的目标，而且是整个WSC系统，包括电力分配和冷却基础设施的目标。 图5.9显示了

几个服务器电源供应器测试得出的效率曲线，也表明峰值功率的比例不足30%。因为

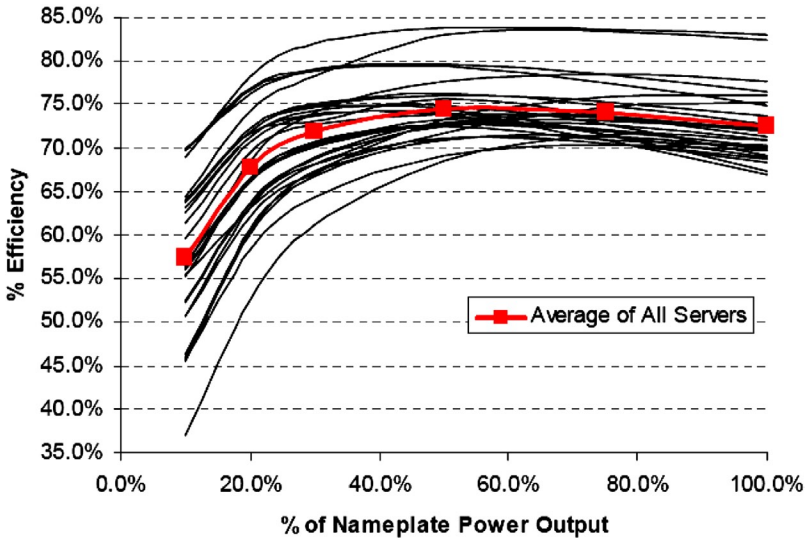

图5.9：几种服务器电源供应器的转换效率调查（由Ton和Fortenbury[84]提供）。

电源通常具有比相应的计算设备更大的峰值容量，因此它们通常在远低于峰值效率评级的情况下运行是很正常的。

5.4 低功耗模式的相对有效性

正如前面讨论的，存在长时间的空闲间隔将使得通过使用各种睡眠模式可以实现更高的能源比例。 我们将这些低功耗模式称为非活动模式，因为设备在这些模式下不可用，而且当负载重新应用时通常会产生相当大的延迟和能源惩罚。 非活动低功耗模式最初是为移动和嵌入式设备开发的，在该领域非常成功。 然而，大多数这些技术在WSC系统中并不适用，因为它们会频繁产生非活动到活动的延迟和能源惩罚。 在这个领域中可以成功的少数技术是具有非常低唤醒延迟的技术，就像CPU的低功耗停机状态（如×86的C1E状态）正在开始出现的情况一样。

不幸的是，这些模式通常也是能源节约程度最小的低功率模式。 关闭磁盘驱动器等非活动低功率模式可以实现大幅节能。 关闭的磁盘几乎不消耗能量，但切换到活动模式会产生能量消耗。

与常规访问相比，延迟惩罚高出1,000倍。 启动磁盘盘片会带来更大的能量惩罚。 如此巨大的激活惩罚限制了磁盘关闭模式只能在设备闲置数分钟的情况下使用，而这在服务器中很少发生。

主动低功率模式是指在不需要闲置的情况下以性能代价节约能量的模式。 CPU电压频率缩放是主动低功率模式的一个例子，因为它仍然能够执行指令，尽管速度较慢。 （目前不可用的）以较低转速读写磁盘驱动器的能力是这类低功率模式的另一个例子。 与非活动模式相比，主动模式在延迟和能量转换到高性能模式的惩罚较大时仍然有用。 由于主动模式是可操作的，系统可以在低能量状态下保持，只要它们保持在某些负载阈值以下。 考虑到低活动期比完全闲置期更常见且更长，切换到主动节能模式的开销更有效地分摊。

5.5 软件在能源比例性中的作用

我们已经提出，硬件组件必须在能源比例性方面进行重大改进，以实现更加节能的WSC系统。 然而，更智能的电源管理和调度软件基础设施在这个领域中起着重要作用。 对于某些组件类型来说，实现完美的能源比例行为可能是不可实现的目标。设计师将不得不实施软件策略，以智能地利用现有硬件的电源管理功能，使用低开销的非活动或主动低功耗模式，并实施任务的节能调度，以提高硬件系统的能源比例性。 例如，如果非活动低功耗模式中的激活惩罚可以足够小，那么可以使用PowerNap文章中描述的类似技术（Meisner等人[55]）来实现只支持非活动低功耗模式的组件的能源比例行为。

这个软件层必须克服两个关键挑战：封装和性能鲁棒性。 能源感知机制必须封装在较低级别的模块中，以最小化向应用程序开发人员公开额外的基础设施复杂性；WSC应用程序开发人员已经处理了前所未有的规模和平台级复杂性。 在大规模系统中，完成一个最终用户任务也往往取决于大量系统以适当的水平执行。 如果个别服务器由于电源管理机制而开始表现出过高的响应时间变异性，那么对服务水平的影响潜力相当高，并且可能导致服务需要额外的机器资源，从而几乎没有改进。

5.6 数据中心电源供应

能效优化自然与较低的电费相关联。 然而，另一个与能源相关的成本因素有时比电费本身更重要：为一组服务器提供一定功率水平的数据中心设施建设成本（也称为电源供应成本）。 根据Uptime Institute每部署的IT瓦特的电源供应成本范围为10-22美元和典型的10年折旧周期，每瓦特IT功率的年度供应成本在1.00-2.20美元的范围内。

相比之下，每瓦IT功耗的相应电费约为1.20美元，假设美国商业用电平均成本为每千瓦时不到0.07美元，PUE系数为2.0。这种成本结构的一个含义是，在机器级别的能效投资节电所带来的电费节省可能是电力供应成本的两倍。 另一个结果是在设施中最大化的电力供应预算的重要性。 例如，如果一个设施的平均功率容量利用率为50%，每瓦使用的供应成本将翻倍。

最大化可用电力预算的使用对于现有设施也很重要，因为它可以让计算基础设施增长或进行升级，而无需获取新的数据中心容量，这可能需要数年时间，如果涉及新建筑。充分利用数据中心的电力预算的动机受到超过其最大容量的业务风险的抵消，这可能导致停机或昂贵的服务协议违规。

5.6.1 部署和电力管理策略

确定正确的部署和电源管理策略需要了解随时间变化的数百或数千台机器组的同时功耗特性。
这个问题复杂化了三个重要因素：

1. 计算设备的额定最大功率（或铭牌值）通常过于保守，因此有限的实用性。
2. 服务器实际消耗的功率随活动量的增加而变化很大，很难预测。
3. 不同的应用程序以不同的方式使用大规模系统。

由于这些因素，最大化数据中心的功耗涉及三个主要步骤：

1. 测量或以其他方式准确估计所有计算设备的实际功耗范围。 主要的系统供应商，如戴尔和惠普，提供在线功耗计算器[21] [46]，如果无法测量，这些计算器可能很有用。

2. 根据应用程序资源使用配置文件，确定在仍然支持峰值使用的最小机器占用空间中合并工作负载的机会。 集群调度程序和虚拟机技术，如VMware [89]和Xen [4]，可以促进这个过程。

3. 了解大型服务器群的同时功耗的统计特性，以揭示通过过度订阅设施电力来部署更多计算机的机会。 服务器群并不总是同时运行在它们的峰值功耗水平上。

   服务器群越大，应用程序的多样性越高，同时出现非常高活动的时间越少。

5.6.2 过度订阅设施电力的优势

过度订阅设施电力包括托管一定数量的系统（以及存储、网络等），其累积峰值功耗将超过设施的最大IT电力预算。 成功实施功耗过度订阅将实际增加数据中心能源预算的整体利用率，并能够托管更多的服务器，同时使超载情况极不可能发生。 我们将对这个问题进行扩展，因为它在技术出版物中得到的关注比前两个步骤要少得多，并且在实践中是一个非常真实的问题[53]。

Fan等人 [27] 研究了通过分析Google在6个月内运行各种工作负载的多达5,000台服务器的功耗行为，来研究过载供电设施的潜在机会。 他们的一个关键结果总结在图5.10中，该图显示了80台服务器（机架）、800台服务器（PDU）和5,000台服务器（集群）的功耗随时间的累积分布。

功耗被归一化为相应组的峰值总功耗。 例如，该图显示，尽管机架单元在80%的时间内使用的功耗低于其峰值功耗的65%，但在6个月的观察窗口期间，它们的功耗曾达到峰值功耗的93%。 对于功耗供应，这表明机架级别的过载供电机会非常低，因为机架可用功耗仅有7%被浪费。 然而，随着机器组规模的增大，情况发生了变化。 特别是整个集群从未超过其总峰值功耗的72%。 因此，如果我们为集群分配的功耗容量与所有机器的峰值功耗之和相对应，那么其中28%的功耗将被浪费。

这意味着在这个功率容量内，我们可以托管近40%的机器。

该研究还评估了更多能量比例的机器在设施级别上减少峰值功耗的潜力。 研究表明，将空闲功耗从峰值的50%降低到10%（即从图5.6中的红色曲线到绿色曲线）可以进一步减少集群峰值功耗超过30%。 这将相当于设施托管能力增加了40%+。

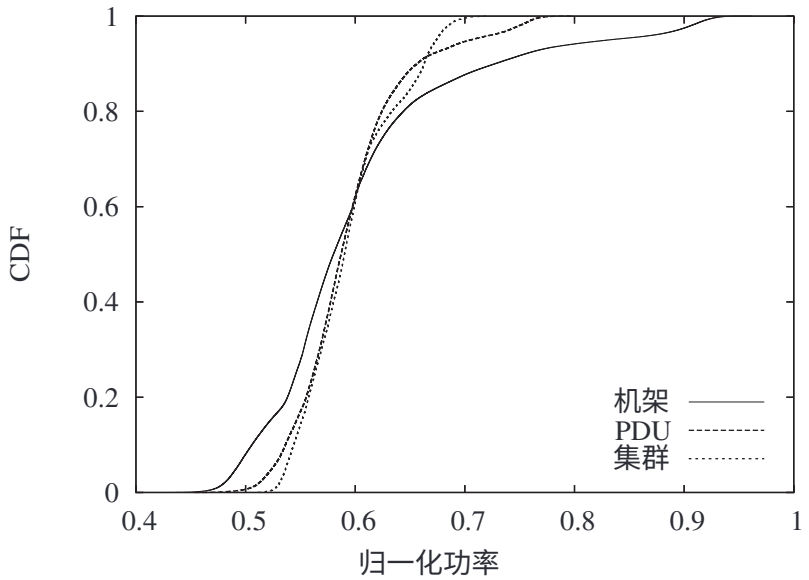

图**5.10**：机器组在给定功率水平（功率水平归一化为相应组合的最大峰值总功率）下所花费的时间的累积分布（**F**an等人[27]）

该研究还发现，在集群中混合不同的工作负载增加了功耗超额分配的机会，因为这降低了机器之间同步功耗峰值的可能性。 一旦使用了超额分配，系统需要一个安全机制来处理工作负载变化可能导致功耗超过数据中心容量的情况。 这可以通过始终将一部分计算资源分配给运行在较低优先级类别中或者没有严格截止时间要求的工作负载（许多批处理工作负载可能属于这一类别）来实现。 这样的工作负载可以快速暂停或中止以减少设施负载。 配置不应过于激进，以至于触发该机制的频率过高，例如如果超额分配应用于机架级别。

在实际部署中，即使经过仔细的设备功率测量、有效的整合和超额订阅策略的部署，设施电力仍然可能被低效利用，这是由于其他实际考虑因素造成的。 例如，一个设施在初始投入使用时很少完全填满，而是根据业务需求增长的情况进行尺寸设计。 因此，在新设施中，部署和使用功率之间的差距往往更大。 设施电力也可能存在一种碎片化形式。 电力使用可能被滞留，仅仅因为增加一个单位（服务器、机架或PDU）可能会超过该级别的限制。 例如，一个2.5千瓦的电路可能

仅支持四台520瓦的服务器，这将保证该电路的利用率低于17%。 如果一个数据中心的设计使得PDU级峰值容量恰好等于所有电路的峰值容量之和，这样的低利用率会在电力传输链上传播，最终在数据中心层面上真正浪费。

5.7 服务器能源使用趋势

与任何机器设计一样，WSC设计旨在实现高能效，同时在各个子系统的容量和性能之间取得平衡，以匹配目标工作负载的预期资源需求。 尽管对于任何给定的工作负载来说，这个最佳点可能会随时间变化，但广泛的工作负载集合的总体行为往往变化较慢。 认识到这种行为对系统设计提供了有用的输入。 例如，对于一个索引搜索程序来说，最佳的CPU速度与存储容量的比例可能是最好的：过多的存储会使搜索变慢，而过少的存储可能会浪费CPU资源。如果所需比例保持不变，但各个组件的能源效率发展速度不同，能源消耗预算可能会随时间发生显著变化。

我们在过去几年中观察到这种现象，即CPU的能效改进超过了DRAM和磁盘存储的改进。 因此，过去占系统能源预算60%以上的CPU现在通常不到45-50%。 以图5.8中使用的系统作为一个平衡的服务器设计的例子：两个双核x86 CPU，主频2.4 GHz，8 GB DRAM和两个磁盘驱动器。 摩尔定律仍然使得CPU的计算速度每18个月左右增加近两倍（通过增加核心数和微小的频率变化），而功耗几乎相同。 但在过去几年中，DRAM和磁盘技术的发展速度没有与之匹配，如果这种趋势持续下去，一个良好平衡的服务器设计的能源使用可能会被存储子系统所主导。

这种趋势的一个结果是CPU电压频率调节在功耗管理中的效用降低。 图5.11显示了通过在不同计算负载下绘制功耗使用情况来展示同一台服务器使用CPU动态电压调节（DVS）可能实现的功耗节约。 当计算负载低于峰值的三分之二时，通过降低频率到1.8 GHz可以实现约10%的节能（超过该负载水平，应用程序将违反延迟SLA）。当利用率进一步降低到三分之一时，可以通过降低频率到1 GHz实现额外的10%节能。然而，随着负载的继续下降，DVS的收益再次回到最大约10%的水平，这是由于系统级能量比例的缺失所导致的。 尽管10-20%的功耗节约并不可忽视，但考虑到大多数系统今天除了CPU DVS之外没有其他功耗管理控制手段，它们并不特别令人印象深刻。此外，如果CPU在整体系统功耗中继续减少，DVS的影响必然会进一步减小。

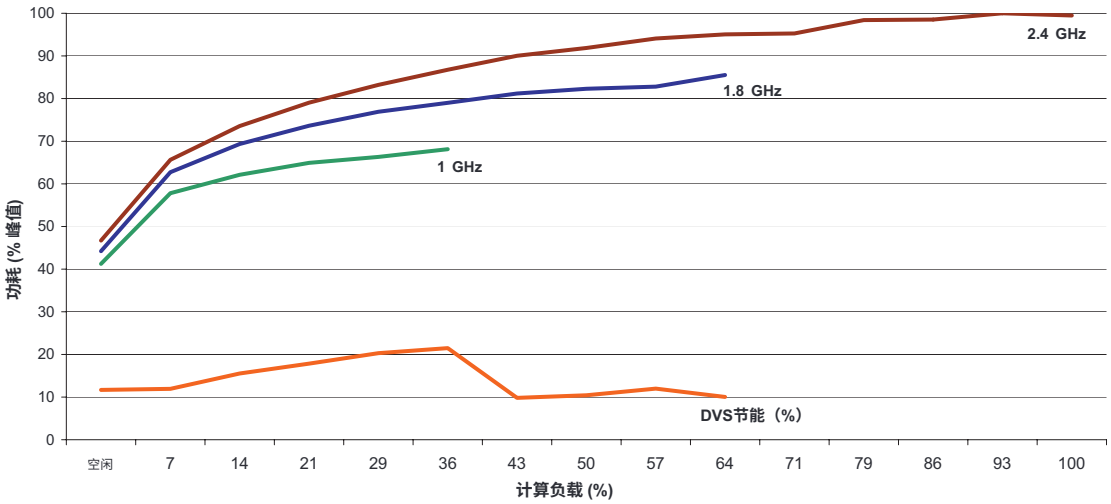

图5.11：x86服务器在三个电压-频率水平下的功率与计算负载关系及相应的能源节约。

5.8 结论

能源效率是WSC的关键成本驱动因素，我们预计能源使用量将成为WSC设计中越来越重要的因素。 目前行业状况不佳：平均实际数据中心和平均服务器效率都太低，主要是因为效率在历史上被忽视，相对于可靠性、性能和资本支出而言，效率一直处于次要地位。 因此，平均WSC浪费了三分之二或更多的能源。

忽视这段历史的好处是，几乎可以轻松获得可观的改进——通过简单地将最佳实践应用于数据中心和服务器设计，可以实现两倍以上的效率改进，风险不大。 不幸的是，超越这些低成本的路径更加困难，需要克服固有复杂问题和不利的技术趋势。 一旦平均数据中心达到最先进的PUE水平，并且已经有高效率电源供应器可用于部署服务器，那么在这些领域进一步提高效率的机会将不到40%。 从研究和开发的角度来看，从现在开始，获得更大的收益机会将需要来自计算机科学家和工程师，而不是机械或电力转换专家。

首先，必须更好地管理电力和能源，以最小化运营成本。 电力决定了整个设施成本，因为很大一部分建设成本与必须支持的最大功率有直接关系。 整体能源使用情况决定了电费以及很大一部分的环境影响。 现今的服务器在实际使用中很少达到高功率，但必须适应或控制（限制）以避免过载。

加载设施的电力传输系统。 功率限制是一种管理服务器池的总功率的有希望的技术，但很难与可用性相协调，即在由突发流量激增或不同集群中的故障引起的紧急情况下需要使用峰值处理能力。 此外，尽管硅门尺寸持续缩小，但峰值服务器功率仍在增加，这是由于操作频率的增加、更大的缓存和内存大小以及更快的片外通信（DRAM和I/O总线以及网络速度）的组合驱动。

其次，现今的硬件不能很好地适应负载条件的变化，因此，在轻负载下，服务器的效率严重降低。 能源比例性承诺为解决这一困境提供了一种方法，但在所有子系统中实施可能具有挑战性——例如，磁盘不自然地适应低功耗活动状态。 用于工作整合的系统可以释放和关闭整个服务器，为非能源比例组件构建的集群创建能源比例行为，但实施和管理更加困难，需要透明的进程迁移，并降低了WSC对突然负载增加的响应能力。 此外，高性能和高可用性的分布式系统软件倾向于以一种减少任何一个系统上足够大的空闲时间的方式来分布数据和计算。 因此，能源管理感知的软件层必须以最小化对性能和可用性的影响的方式制造空闲时间。

最后，能源优化是一个复杂的端到端问题，需要在硬件、操作系统、虚拟机、中间件、应用程序和运营组织之间进行复杂的协调。 即使是小错误也可能破坏能源节约，例如，当次优的设备驱动程序产生过多的中断或者邻近机器的网络通信阻止机器进入静默状态时。 涉及的组件太多，无法自然地实现完美的协调，而且我们目前缺乏管理这种复杂性的正确抽象。 与硬件改进（如可按比例调节能源的组件）不同，解决这个端到端问题的规模将会更加困难。

5.8.1 进一步阅读

能源、峰值功率和温度的管理已成为越来越多研究的目标。Chase等人[14]、G. Chen等人[15]和Y. Chen等人[16]考虑了在数据中心自动配置资源时考虑能源节约和应用程序性能的方案。Raghavendra等人[69]描述了一个综合的数据中心功耗管理框架，通过控制理论方法将硬件级功耗限制与虚拟机调度机制协调起来。 Femal和Freeh[28,29]专注于数据中心功耗过度订阅和动态电压频率调节作为减少峰值功耗的机制。 最后，管理温度是Heath等人[45]和Moore等人[56]提出的系统的主题。

第六章

成本建模

为了更好地理解能源相关优化的潜在影响，让我们来研究一下数据中心的总拥有成本（TCO）。 在顶层，成本分为资本支出（Capex）和运营支出（Opex）。Capex指的是必须提前投资的项目然后在一定时间范围内进行折旧的投资；例如数据中心的建设成本或服务器的购买价格。Opex指的是实际运行设备的经常性月度成本；包括电费、维修和保养费用、现场人员的工资等等。因此，我们有：

$$\text{TCO} = \text{数据中心折旧} + \text{数据中心Opex} + \text{服务器折旧} + \text{服务器Opex}$$

在本章中，我们关注顶线估计，适当简化模型。 更详细的成本模型可以在文献[61,50]中找到。 对于学术目的，我们简化的模型足以模拟所有主要成本；与实际数据中心相比，主要的不准确性将来自于模型输入值，如建设成本。

6.1 资本成本

数据中心的建设成本因设计、规模、位置和所需速度而异。 毫不奇怪，增加可靠性和冗余性会使数据中心更加昂贵，而非常小或非常大的数据中心往往更昂贵（前者是因为固定成本无法分摊到许多瓦特，后者是因为大型中心需要额外的基础设施，如电力变电站）。 表6.1显示了一系列典型数据中心建设成本的范围，以每瓦特可用关键功率的美元表示，这些数据来自各种来源。 根据经验法则，大多数大型数据中心的建设成本可能在每瓦特12-15美元左右，而较小的数据中心成本更高。

以每瓦特的美元来衡量成本对于较大的数据中心是有意义的（其中与规模无关的固定成本在总成本中占比相对较小），因为数据中心的主要组成部分——电力、冷却和空间——与瓦特大致呈线性关系。 通常，总建设成本的大约80%用于电力和冷却，剩下的20%用于一般建筑和场地建设。

表6.1：以美元每瓦特的关键功率表达的数据中心建设成本范围。

成本/瓦特	来源
$12–25	Uptime Institute对中小型数据中心的估计；较低的值适用于很少在实践中使用的“Tier 1”设计。
$10+	微软以2亿美元购买了加利福尼亚州的两个10兆瓦数据中心；此成本不包括土地和建筑物的价值。
$10–14	杜邦法布罗斯的S-1文件（第6页）包含了以下成本信息： 1.024亿美元用于购买ACC4，一个现有的9.6兆瓦设施（每瓦10.67美元） 1.8-3亿美元用于建造四个18.2兆瓦的设施（每瓦10美元—13.40美元） 2008年第三季度的收益公告[70]列出了一个已完成的18.2兆瓦芝加哥设施的建造成本为1.9亿美元，即每瓦10.44美元。

关键功率被定义为可以为IT设备提供的峰值功率水平。

成本随所需冗余和可用性程度而变化，因此我们总是以每个关键瓦特的美元来表示成本，也就是每个实际可以被IT设备使用的瓦特。
例如，一个拥有20兆瓦发电机的数据中心可能是以2N配置建造的，提供的关键功率只有6兆瓦（再加上4兆瓦用于冷却设备的供电）。 因此，如果花费1.2亿美元建造，它的成本是每瓦特20美元，而不是每瓦特6美元。 行业报告经常不正确地使用术语“关键功率”；因此，我们的示例数据中心可能被描述为一个20兆瓦的数据中心，甚至是一个30兆瓦的数据中心，如果它由一个能提供30兆瓦的电力分站供电。经常以每平方英尺的美元报价成本，但这个度量标准不如以每瓦特的美元来比较项目有用，而且使用的一致性比以每瓦特的美元来表示的成本还要不一致。 特别是，在计算中没有关于要包括或排除哪些空间的标准定义，并且这个度量标准与数据中心建设的主要成本驱动因素——关键功率——的相关性不强。
因此，大多数行业专家避免使用每平方英尺的美元来表示成本。

每月折旧费用（或摊销费用）取决于投资的摊销期限（与预期寿命有关）和假定的利率。 通常，数据中心的折旧期为10至15年。根据美国会计准则，常用直线折旧法。

资产价值每个月下降固定金额。 例如，如果我们在12年内对一个每瓦15美元的数据中心进行折旧，折旧成本为每个月0.10美元/瓦。 如果我们不得不以8%的利率贷款来融资建设，相关的每月利息支付额外增加了0.06美元/瓦，总共为每个月0.16美元/瓦。 典型的利率会随时间变化，但许多公司将支付10-13%的利息。

服务器成本的计算方式类似，只是服务器的寿命较短，通常在3-4年内折旧。 为了使服务器和数据中心的成本标准化，使用每瓦特的服务器成本是有用的，以服务器的实际峰值功耗作为分母。 例如，一台价格为4000美元的服务器，实际峰值功耗为500瓦，成本为8美元/瓦特。 在4年内折旧，该服务器每个月的成本为0.17美元/瓦特。 以8%的年利率融资该服务器，每个月增加0.03美元/瓦特，总计每个月0.20美元/瓦特，与数据中心的每瓦特成本相同。

6.2 运营成本

数据中心的运营成本很难确定，因为它严重依赖于运营标准（例如，同时值班的安保人员数量或发电机的测试和维护频率）以及数据中心的规模（较大的数据中心更便宜，因为固定成本更好地摊销）。 成本还可能因地理位置（气候、税收、工资水平等）以及数据中心的设计和年龄而有所不同。为简单起见，我们将运营成本分解为每瓦特的月费，代表安保人员、维护和电费等项目。美国多兆瓦数据中心的典型运营成本范围为每个月0.02美元至0.08美元/瓦特，不包括实际电费成本。

同样，服务器也有运营成本。 因为我们只关注基础设施本身的运行成本，所以我们将重点放在硬件维护和修理以及电力成本上。 服务器维护成本因服务器类型和维护标准（例如，四小时响应时间与两个工作日）而有很大差异。

此外，在传统的IT环境中，大部分运营成本都在应用程序上，即软件许可证和系统管理员、数据库管理员、网络工程师等的成本。我们在这里排除了这些成本，因为我们关注的是物理基础设施的运行成本，但也因为应用程序的成本因情况而异。 在小型企业环境中，每几十台服务器通常会有一个系统管理员，导致每台机器的年度成本相当可观[73]。许多已发表的研究试图量化管理成本，但其中大部分都是由试图证明其产品成本效益的供应商资助的，因此可靠的无偏信息很少。 然而，通常认为大规模应用程序需要较少的管理，可能每个管理员可扩展到1,000台服务器。

6.3 案例研究

考虑到涉及的变量数量较多，最好通过研究一小部分代表不同部署类型的案例来说明成本因素的范围。首先，我们考虑一个典型的新型多兆瓦数据中心在美国（更接近Uptime Institute的Tier 3分类），该数据中心完全配备了2008年仍然可以被认为是2008年体积可安装的服务器产品的高端服务器。对于这个例子，我们选择了一台Dell 2950 III EnergySmart，配备16 GB的内存和四个硬盘，根据Dell的数据中心容量规划工具，在峰值时每台服务器的功耗为300瓦特，价格约为6000美元（标价，截至2008年）。其余的基准参数选择如下：

- 电力成本为2006年美国工业平均电价6.2美分/千瓦时。
- 企业必须支付的贷款利率为12%，我们用3年的仅利息贷款来融资服务器。

- 数据中心建设成本为每瓦15美元，在12年内分摊。
- 数据中心的运营费用为每瓦每月0.04美元。
- 数据中心的功耗效率（PUE）为2.0。
- 服务器的寿命为3年，服务器的维修和保养费用为资本支出的5%每年。
- 服务器的平均功耗为峰值功耗的75%。

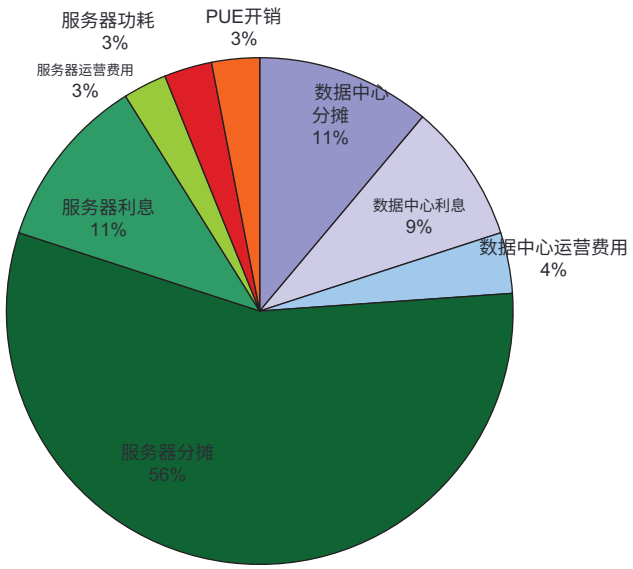

图**6.1**：案例研究A的TCO成本分解。

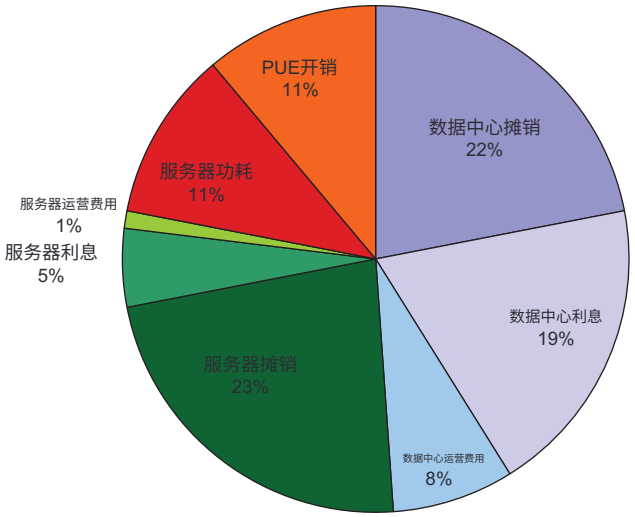

图**6.2**：案例研究B（成本较低，功耗较高的服务器）的TCO成本分解。

图6.1显示了案例A的年度TCO在数据中心和服务器相关的运营费用和资本支出组成部分之间的分解。[1]

在这个例子中，这是典型的经典数据中心，高服务器资本成本占据了整体TCO的主导地位，其中69%的月度成本与服务器的购买和维护有关。 然而，基于商品的低成本（也许可靠性较低）服务器，或者更高的电力价格，可以极大地改变情况。 对于案例B（参见图6.2），我们假设一台更便宜、更快、功耗更高的服务器，在峰值时消耗500瓦特，成本仅为2000美元，在电费为0.10美元/千瓦时的地方。在这种情况下，数据中心相关成本占总成本的49%，能源成本占22%，服务器成本下降至29%。 换句话说，在这种情况下，这样一台服务器的托管成本，也就是所有基础设施和供电的成本，是购买和维护服务器成本的两倍多。

请注意，即使假设电力价格更高且服务器功耗更高，情况B的绝对3年总拥有成本比情况A更低（$8,702对$10,757），因为服务器更便宜。 与电力相关成本的相对重要性可能会增加，如情况B所示

1本节中图表的电子表格在线版本可在http://spreadsheets.google.com/pub?key=phRJ4tNx2bFOHgYskgpoXAA&output=xls找到。

因为CPU的功耗（和性能）在1995年至2007年间增加了8倍，大约每年增长19% [82]，而低端服务器的销售价格相对稳定。 因此，服务器硬件的每瓦特成本正在下降，而电力和建筑成本正在上升。这些趋势的结果是明显的：越来越多，服务器的总成本将主要取决于其功耗，而服务器的购买价格将变得不那么重要。 换句话说，从长远来看，数据中心设施成本（与功耗成正比）将占据总成本的越来越大的比例。

我们并不预测未来一定会像这样发展，但如果不改变，情况就会如此。

6.3.1 真实世界的数据中心成本

事实上，实际的数据中心成本甚至比目前的模型更高。到目前为止，所有提出的模型都假设数据中心是100%满载，并且服务器相当繁忙（75%的峰值功耗对应于大约50%的CPU利用率；请参见第5章）。 实际上，情况经常并非如此。 例如，由于数据中心空间需要一段时间来建设，我们可能希望保留一定的空余空间以容纳未来的部署。 此外，服务器布局假设功耗过高（最坏情况）。 例如，一台服务器可能在安装了所有选项（最大内存、磁盘、PCI卡等）时消耗“高达”500瓦特，但实际部署的配置可能只使用300瓦特。如果服务器布局假设“名牌”额定功率为500瓦特，我们只能达到60%的利用率因子，因此实际数据中心每台服务器的成本增加了1.66×。 因此，实际上，每台服务器的实际月成本往往比上述显示的要高得多，因为与数据中心相关的成本与数据中心功耗成反比增加。

正如第5章所讨论的那样，实现高数据中心功耗利用率并不像看起来那么简单。 即使供应商提供了一个功耗计算器来计算特定配置的实际最大功耗，该值也假设100%的CPU利用率。 如果我们根据该值安装服务器，并且它们平均只运行30%的CPU利用率（消耗200瓦而不是300瓦），我们就浪费了30%的数据中心容量。 另一方面，如果我们根据平均值200瓦进行安装，而在月底服务器实际上在一段时间内运行接近满负荷，我们的数据中心将过热或跳闸。 同样，如果我们在以后的时间选择给服务器添加额外的内存或磁盘，这将需要对服务器机架进行物理解压缩，如果我们的功耗计算中没有留下任何余地。 因此，在实践中，数据中心运营商会留下相当多的余地来防范这些问题。储备量为20-50%。

这意味着实际数据中心很少以接近其额定容量的任何地方运行。
换句话说，一个拥有10兆瓦关键功率的数据中心通常每月平均消耗4-6兆瓦的实际关键功率（加上PUE开销）。

6.3.2 对部分填充的数据中心进行建模

要对部分填充的数据中心进行建模，我们只需将数据中心的Capex和Opex成本（不包括电力）按照占用因子的倒数进行缩放（图6.3）。例如，一个只有三分之二满的数据中心的Opex成本增加了50%。以案例B为例，但占用因子为50%，数据中心成本完全主导了成本（参见图6.2），只有总成本的19%与服务器相关。 考虑到刚刚讨论的备用电力需求，这种情况并不像听起来那么牵强。
因此，改善实际数据中心的使用情况（例如使用功率限制）可以大幅降低实际数据中心的成本。 以绝对美元计算，在一个完全满载的数据中心中，服务器的总拥有成本为8,702美元，而在一个半满的数据中心中为12,968美元，而这个服务器的购买成本只有2,000美元

部分使用的服务器也以积极的方式影响运营成本，因为服务器使用的功率较少。 当然，这些节省是有问题的，因为在这些服务器上运行的应用程序很可能产生较少的价值。我们的TCO模型无法捕捉到这种效应，因为它仅基于物理基础设施的成本，而不包括运行在其上的应用程序。

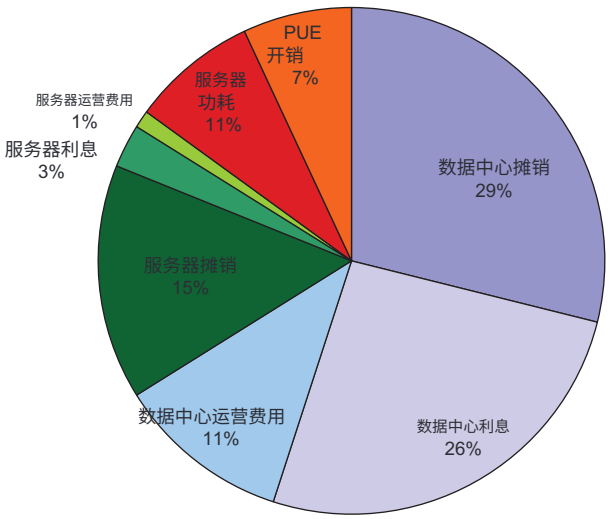

图**6.3**：TCO案例研究C（部分填充设施）。

硬件。 为了衡量这种端到端的性能，我们可以测量一个应用价值的代理（例如，完成的银行交易数量或Web搜索数量）并将TCO除以该值。 例如，如果我们有一个每月花费100万美元且完成1亿次交易的数据中心，每次交易的成本将为1美分。 另一方面，如果一个月的流量较低，我们只完成了5000万次交易，每次交易的成本将翻倍至2美分。 在本章中，我们专注于硬件成本，但重要的是要记住，最终软件性能和服务器利用率同样重要。

第7章

处理故障和维修

只有当用户能够相信他们越来越依赖的服务始终可用时，基于Web的面向服务的计算的承诺才能完全实现。 这种可用性期望转化为对建筑规模计算机的高可靠性要求。 确定适当的可靠性水平基本上是在故障成本（包括维修）和预防故障成本之间的权衡。 对于传统服务器来说，故障成本被认为非常高，因此设计师们不遗余力地提供更可靠的硬件，例如添加冗余电源、风扇、错误校正编码（ECC）、RAID磁盘等。许多传统企业应用程序在经常发生硬件故障后无法设计成容错的，事后很难使它们具备容错能力。 在这种情况下，使硬件非常可靠成为一种合理的选择。

然而，在规模上，仓库级计算机（WSCs）的硬件不容易变得“足够可靠”。 假设一个集群具有超可靠的服务器节点，平均故障间隔时间（MTBF）为30年（10,000天）-远远超出以实际成本实现的典型可能性。 即使有这些理想可靠的服务器，一个由10,000个服务器组成的集群每天平均会出现一个服务器故障。因此，任何需要整个集群正常运行的应用程序的MTBF都不会超过1天。实际上，典型服务器的MTBF远远低于30年，因此实际集群的MTBF将在几小时到故障之间。 此外，大型复杂的互联网服务通常由几个软件模块或层组成，这些模块或层本身可能会以比硬件组件更高的故障率发生故障。 因此，WSC应用程序必须通过软件解决故障服务器的问题，可以通过应用程序本身的代码或通过中间件提供的功能来实现，例如用于虚拟机的供应系统，在备用节点上重新启动失败的虚拟机。 Hamilton [43]根据在MSN和Windows Live上设计和运营一些最大的服务的经验，讨论了在这种环境中编写软件的一些影响。

7.1 基于软件的容错的影响

WSC中故障的不可避免性使得容错软件本质上比可以假设无故障操作的软件更加复杂。尽可能地，应该尝试实现

一个容错软件基础架构层可以将大部分故障复杂性从应用级软件中隐藏起来。

然而，采用这种模型也有一些积极的后果。 一旦硬件故障可以在不对服务造成过多中断的情况下被容忍，计算机架构师就有一定的余地来选择最大化整体系统成本效益的硬件可靠性水平。 这种余地使得可以考虑使用廉价的PC级硬件作为服务器平台，而不是大型机级计算机，如第3章所讨论的。 此外，这种模型还可以简化常见的操作流程。 例如，要升级集群中的系统软件，可以在后台加载新版本（即在正常运行期间），终止旧版本，并立即启动新版本。硬件升级也可以遵循类似的过程。 基本上，用于处理服务器故障的相同容错软件基础架构机制可以具备支持广泛操作流程的所有所需机制。

通过选择适当的时间窗口和限制杀死-重启操作的速度，操作员仍然可以管理所需的计划服务级别中断的数量。

这里利用的基本属性是，与传统的服务器设置不同，不再需要不惜一切代价保持服务器运行。 这个简单的要求转变几乎影响了部署的方方面面，从机器/数据中心设计到运营，通常可以实现其他情况下无法实现的优化机会。 例如，让我们来看看这对恢复模型的影响。 在不可避免的瞬态硬件故障（如宇宙粒子撞击引起的不可纠正错误）存在的情况下，需要高可靠性的系统可能需要硬件支持进行检查点恢复，以便在检测到故障时可以从先前的正确状态重新启动执行。 允许在发生此类故障时关闭的系统可能选择不承担检查点的额外开销和能源消耗。

另一个有用的例子涉及可靠存储系统的设计权衡。 一种选择是通过使用多个磁盘驱动器在镜像或RAID配置中构建高度可靠的存储节点，以便可以在运行时纠正多个磁盘错误。磁盘冗余性增加了可靠性，但本身并不能保证存储服务器始终正常运行。还需要解决许多其他单点故障（电源供应、操作系统软件等），处理所有这些问题会增加额外成本，但永远不能保证无故障运行。 另一种选择是将数据镜像或RAID在多台机器上的磁盘驱动器上，这是Google的GFS [31]所选择的方法。 这种选择不仅容忍磁盘故障，还容忍整个存储服务器崩溃，因为每个数据片段的其他副本可以通过其他服务器访问。 它还具有与集中式存储服务器场景不同的性能特征。 数据更新可能会产生更高的网络开销，因为它们需要与多个系统通信以更新所有副本，但聚合读取带宽可以大大增加，因为客户端可以从多个终端获取数据（在完全复制的情况下）。

在一个可以容忍多个软件级别故障的系统中，对硬件层的最低要求是其故障总是能够及时检测并报告给软件，以便软件基础设施能够对其进行控制并采取适当的恢复措施。并不一定要求硬件能够透明地纠正所有故障。 这并不意味着这类系统的硬件应该设计成没有错误纠正能力。

只要纠错功能在成本或复杂度上是合理的，通常值得支持。这意味着如果硬件纠错成本过高，系统可以选择使用仅提供检测能力的更便宜版本。 现代DRAM系统是一个很好的例子，它可以以非常低的额外成本提供强大的纠错能力。

然而，放宽对硬件错误检测的要求将更加困难，因为这意味着每个软件组件都需要负担起检查自身正确执行的任务。 在其历史的早期阶段，Google不得不处理缺乏奇偶校验的DRAM服务器。 生成Web搜索索引基本上是一个非常大的洗牌/合并排序操作，使用多台机器在很长的时间内进行。 在2000年，当一部分测试查询返回看似随机的文档时，Google的Web索引的每月更新都没有通过预发布检查。 经过一些调查，发现了新索引文件中对应于数据结构中某个位置被卡住为零的位的模式；这是通过一个有故障的DRAM芯片流式传输大量数据的副作用。 为了最大程度地减少此类问题的发生，索引数据结构中添加了一致性检查，并且没有报告此类问题的进一步情况。 然而，请注意，这种解决方法并不能保证在索引过程中100%的错误检测，因为并没有检查所有的内存位置，例如指令就没有被检查。它之所以有效，是因为索引数据结构比计算中涉及的所有其他数据都要大得多，这样具有缺陷的DRAM的机器很可能被识别并从集群中排除。 Google的下一代机器确实包括内存奇偶校验检测，并且一旦带有ECC的内存价格降至具有竞争力的水平，所有后续的机器都使用了ECC DRAM。

7.2 故障分类

高效的容错软件层必须基于一组关于故障来源、它们的统计特性和相应恢复行为的期望。在没有这些期望的情况下开发的软件，如果底层故障被低估，就容易发生故障，如果假设故障比实际生活中更频繁，就需要过度配置。

在不同部署中，WSC系统的设备和软件基础设施的多样性使得准确量化故障成为一项具有挑战性的任务。 相反，我们将尝试从公开可用的来源和我们自己的经验中总结高级别的趋势。

7.2.1 故障严重程度

硬件或软件故障可能以不同程度影响互联网服务，导致不同的服务级别故障模式。 最严重的模式可能需要非常高的可靠性水平，而最不严重的模式可能具有更宽松的要求，可以通过更低成本的解决方案实现。 我们将服务级别故障广泛分类为以下几类，按严重程度递减排列：

- 损坏：无法再生、丢失或损坏的已提交数据
- 无法访问：服务已停止或用户无法访问
- 降级：服务可用，但以某种降级模式提供
- 掩盖：故障发生，但通过容错软件/硬件机制完全隐藏对用户的影响。

在这些类别中，可接受的鲁棒性水平会有所不同。 我们预计大多数故障都会被精心设计的容错基础设施所掩盖，以使其在服务提供商之外基本上是不可见的。 掩盖的故障可能会影响服务的最大可持续吞吐能力，但通过谨慎的过度配置可以确保服务保持健康。

如果故障无法完全掩盖，其最轻微的表现形式是服务质量的某种程度的降级。 在这种情况下，不同的服务可以以不同的方式引入降级可用性。 Brewer [10] 提出的一种故障类别示例是，Web搜索系统使用数据分区技术来提高吞吐量，但会丢失一些服务数据库的部分。 在许多情况下，搜索查询结果可能不完美，但仍然可以接受。 由于故障而导致的优雅降级还可以通过降低数据的新鲜度来体现。 例如，用户可能可以访问他或她的电子邮件账户，但新邮件的投递可能会延迟几分钟，或者邮箱中的某些片段可能会暂时丢失。 尽管这些故障也需要尽量减少，但它们比完全不可用的情况要轻微。 互联网服务需要有意地设计以充分利用这种优雅降级的机会。 换句话说，这种支持通常非常应用程序特定，而不是轻易隐藏在集群基础设施软件的各个层次之中。

服务的可用性/可达性非常重要，特别是因为互联网服务的收入通常与流量有关 [12]。 然而，完美的可用性并不是互联网连接服务的现实目标，因为互联网本身的可用性特征有限。Chandra等人 [91] 报告称，由于各种连接问题，包括路由问题，互联网终端可能在1%至2%的时间内无法相互连接。 这意味着可用性不到两个“九”。换句话说，即使您的互联网服务完全可靠，用户平均来说也只能感知到不超过99.0%的可用性。 因此，一个避免长时间中断对任何大型用户群体产生影响且平均不可用性低于1%的互联网连接服务将很难与完全可靠的系统区分开来。 谷歌对互联网可用性的测量表明，当谷歌服务器是其中一个终端时，平均可用性可能不超过99.9%，但范围相当广泛。 世界上某些地区的可用性明显较低。

以绝对时间来衡量服务的可用性对于通常出现大量每日、每周和季节性流量变化的互联网服务来说不太有用。 一个更合适的可用性指标是由服务满足的请求的比例除以用户发出的总请求数量；这个指标被 Brewer [10] 称为产出。

最后，一类特别有害的故障是关键数据、特别是用户数据、关键操作日志或难以或不可能再生的相关数据的丢失或损坏。 可以说，对于服务而言，不丢失数据比对所有用户完全可用更为重要。 还可以认为这样的关键数据可能只占给定服务操作中所涉及的所有数据的相对较小部分。 例如，Web 的副本及其对应的索引文件是庞大而重要的搜索引擎数据，但最终可以通过重新爬取丢失的分区和重新计算索引文件来再生。

总结一下，在互联网服务中并不普遍需要接近完美的可靠性。 虽然对于关键数据损坏等故障来说，实现完美的可靠性是可取的，但大多数其他故障模式可以容忍较低的可靠性特征。 由于互联网本身的可用性不完美，用户可能无法察觉到完全可用的服务与具有4个九的可用性之间的服务质量差异。

7.2.2 服务级别故障的原因

在WSC中，了解故障对整个系统健康的影响是有用的，例如造成停机或其他严重的服务级别中断。 Oppenheimer等人[58]研究了三个由500多台服务器组成的互联网服务，并试图确定服务级别故障的最常见来源。 他们得出结论，操作员引起的或配置错误是服务级别故障的最大贡献者，而硬件相关故障（服务器或网络）占总故障事件的10-25%。

THEDATACENTERASACOMPUTEr错误是服务级别故障的最大贡献者，而硬件相关故障（服务器或网络）占总故障事件的10-25%。

Oppenheimer的数据与Gray的开创性工作[35]在某种程度上一致，后者不是研究互联网服务，而是分析了1985年至1990年间高度容错的Tandem服务器的实际数据。 他还发现硬件故障只占总故障的一小部分（不到10%）。 软件故障（约60%）和维护/操作故障（约20%）占据了故障统计数据的主导地位。

最初看到这两个完全不同的系统中硬件故障对故障事件的贡献如此之少，有点令人惊讶。 这些数字并不是在说明这些系统中硬件组件的可靠性，而是表明容错技术在防止组件故障影响高级系统行为方面取得了多大的成功。 在Tandem的情况下，这些技术主要是通过硬件实现的，而在Oppenheimer研究的系统中，我们可以归功于容错软件基础设施的质量。 无论是基于软件还是硬件的容错技术，在故障主要是统计独立的情况下表现特别好，这在硬件故障中通常（即使不总是）是成立的。 可以说，软件、操作员和维护引起的故障对故障的影响较大的一个重要原因是，它们更有可能同时影响多个系统，从而创建一个相关故障场景，这更难以克服。

我们在谷歌的经验通常与Oppenheimer的分类相一致，即使类别定义不完全一致。 图7.1表示了所有事件的粗略分类

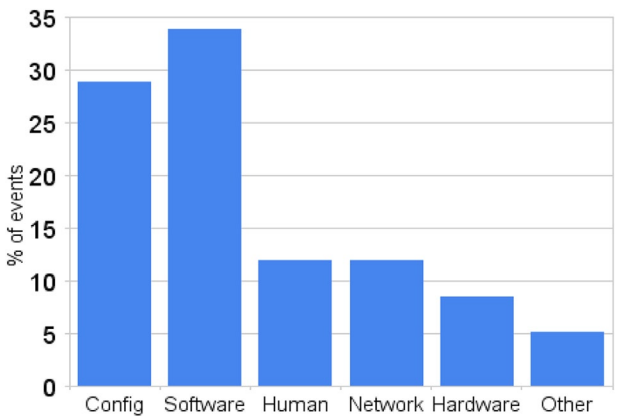

图**7.1**：谷歌主要服务中最可能导致服务中断事件的分布（初步数据，仅为谷歌的Robert Stroud提供的6周数据）。

这些对应于谷歌的大规模在线服务中服务级别的明显中断。 这些不一定是停机（实际上，其中大部分甚至不是用户可见的事件），但它们对应于监控基础设施注意到某种服务降级并需要运维团队进行详细检查的情况。 正如预期的那样，与软件错误、错误的配置数据和人为错误相比，机器或网络故障更不可能导致服务中断。

7.3 机器级别的故障

在设计容错分布式系统时，理解服务器级别的可用性是一个重要因素。 在这里，我们将机器级故障定义为导致服务器宕机的所有情况，无论原因是什么（例如，包括操作系统错误）。

与集群服务故障一样，关于服务器可用性的已发表的实地数据相对较少。 Kalyanakrishnam等人在1999年的一项研究中发现，商业组织的邮件路由服务中的Windows NT机器平均可用性为99%。 作者们观察到66台服务器上发生了1,100次重启事件，平均正常运行时间为11.82天（中位数为5.54天），平均停机时间不到2小时（中位数为11.43分钟）。 约一半的重启被归类为异常，即由于系统问题而不是正常关机。 只有10%的重启可以归咎于故障硬件或固件。 数据表明，应用程序故障、连接问题或其他系统软件故障是已知的最常见的崩溃原因。 如果我们只关注被归类为异常的重启事件，我们得到的平均无故障时间（MTTF）约为22天，或者年化机器故障率超过1600%。

Schroeder和Gibson [75]研究了洛斯阿拉莫斯国家实验室高性能计算系统的故障统计数据。虽然这些计算机不是我们在这里感兴趣的类型，但它们由类似于WSC中的单个服务器的节点组成，因此它们的数据对于理解我们环境中的机器级故障是相关的。 他们的数据涵盖了近24,000个处理器，其中超过60%部署在小规模SMP（每个节点2-4个处理器）的集群中。 尽管节点故障率在不同系统之间变化超过10倍×，但按处理器数量归一化的故障率要稳定得多，大约为每个CPU每年0.3个故障，这表明插座数量与不可靠性之间存在线性关系。 如果我们假设服务器有四个CPU，我们可以预期机器级故障率约为每年1.2个故障或MTTF约为10个月。 这种服务器故障率比Kalyanakrishnam的研究观察到的故障率低了14倍以上！[2]

[2]作为参考，在英特尔技术简报[61]中，英特尔数据中心的服务器故障率报告为10个月的故障率为3.83%，相当于年化故障率约为4.6%。

谷歌的机器级故障和停机统计数据总结在图7.1和图7.2中（由谷歌的安德鲁・摩根提供）。这些数据基于对所有机器重新启动事件及其相应停机时间的6个月观察，其中停机时间指的是机器不可用于服务的整个时间间隔，无论原因如何。 这些统计数据涵盖了谷歌的所有机器。 例如，它们包括处于维修流程中的机器，计划中的升级停机时间，以及各种机器崩溃。

图7.2显示了机器重新启动事件的分布。 超过一半的服务器在观察期内保持运行，超过95%的机器重新启动频率不到一个月一次。 然而，尾部相对较长（图中截断了11次或更多次重新启动的数据）。
例如，大约1%的机器重启频率超过一周一次。

然而，这种大规模平均化会模糊掉一些效应。 例如，在新服务器产品推出的前几个月，我们通常会看到高于正常的故障率。 造成这种情况的原因包括制造引导效应、固件和内核错误，以及只有在大量系统使用后才会变得明显的偶发硬件问题。如果我们从样本中排除仍然受到这些影响的机器，年化重启率将下降约2倍，平均每次重启间隔超过5个月。我们还注意到，重启频率非常高的机器很少能够长时间处于活动状态。 如果我们排除重启频率超过一周一次的机器（约占总数的1%），平均年化重启率将降至2.53。

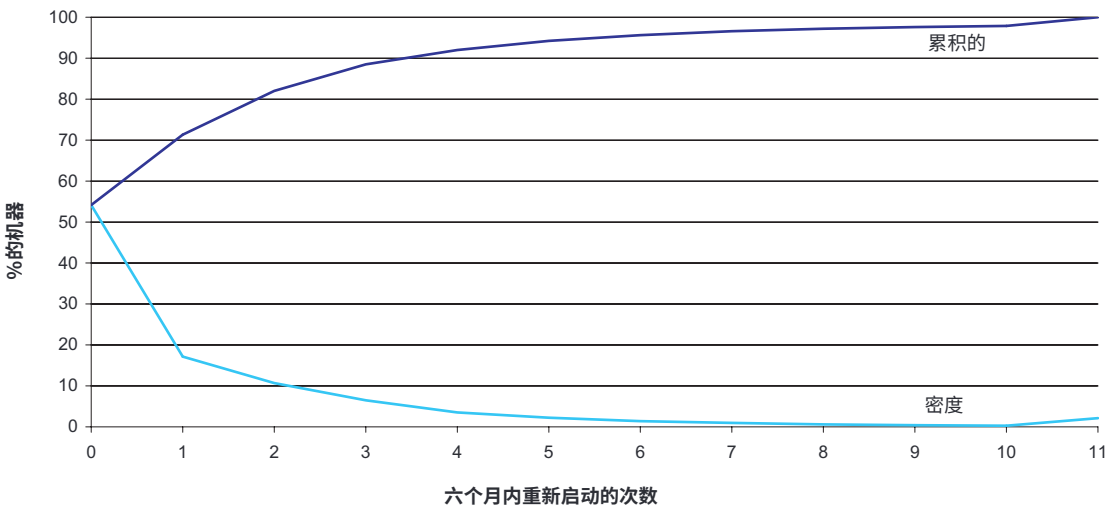

图**7.2**：**G**oogle在6个月内机器重新启动的分布。

重新启动统计数据是容错软件系统设计的关键参数，但是只有将其与停机数据结合起来，可用性情况才能完整地呈现出来，这一点早在伯克利ROC项目[62]中已经明确提出。 图7.3显示了来自Google服务器相同群体的停机时间分布。 *x*轴显示了停机时间，同时显示了密度和累积机器分布。 请注意，数据包括计划重新启动和由各种硬件和软件故障引起的重新启动。 停机时间包括机器停止运行直到重新启动和基本节点服务重新启动的所有时间。 换句话说，停机时间间隔不是在机器完成重新启动时结束，而是在关键的基本守护进程启动时结束。

大约55%的重启事件持续时间不超过6分钟，其中25%的事件持续时间在6到30分钟之间，剩下的大部分重启事件在一天左右完成。大约1%的重启事件持续时间超过一天，这可能对应系统进行维修。 平均停机时间略长于3小时，这个值受到持续时间在30到2,000分钟（约33小时）之间的重启事件的影响。 这些慢重启的原因有多种，包括文件系统完整性检查、需要半自动重启过程的机器挂起以及机器软件重新安装和测试。 由此得出的平均机器可用性为99.84%，接近“三个九”。

在提供容错软件系统时，重要的是专注于真实（意外的）机器崩溃，而不是上述分析考虑的所有重启事件。根据我们的经验，

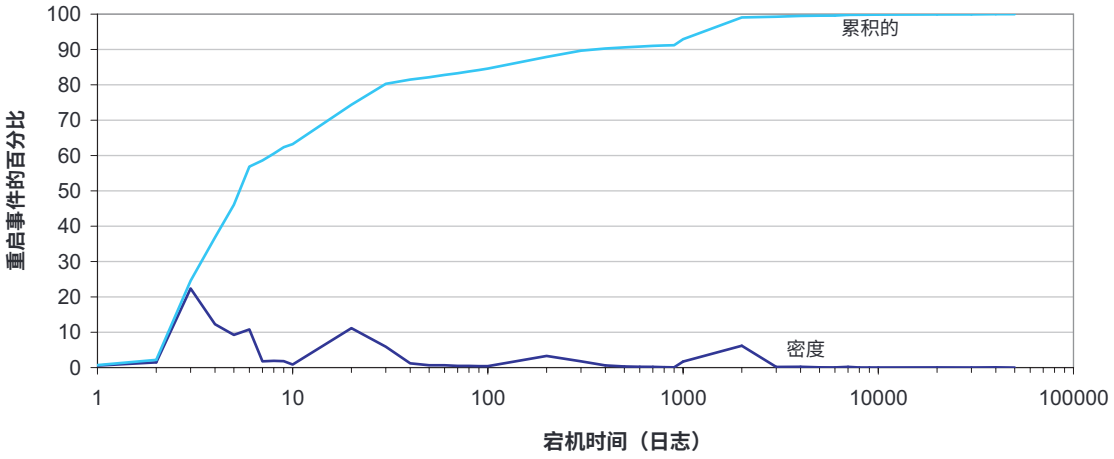

图**7.3**：在**G**oogle观察到的机器宕机分布，持续6个月。 所有机器的平均年化重启率为4.2，对应着平均重启间隔不到**3**个月。

成熟服务器（即经历过婴儿期的服务器）的崩溃率在每年1.2到2次之间。 实际上，这意味着使用2,000台服务器的服务应该计划在正常运行时容忍大约每2.5小时发生一次机器崩溃，或者每天大约10台机器。考虑到99%的重启情况下预期的机器宕机时间不到2天，只需要20台备用机器就足以安全地保持服务的完全供应。 如果有大量必须加载的状态以使机器准备好提供服务，则可能需要更大的余量。

7.3.1 什么导致机器崩溃?

可靠地识别机器崩溃的罪魁祸首通常很困难，因为有许多情况下，瞬态硬件错误很难与操作系统或固件错误区分开来。然而，有大量间接和轶事证据表明，由软件引起的崩溃比硬件故障引起的崩溃要常见得多。 其中一些证据来自于组件级诊断。 因为内存和磁盘子系统故障是2007年谷歌发送到硬件维修的服务器中最常见的诊断问题，所以我们将重点关注这两个问题。

DRAM软错误。虽然关于这个问题的现场数据很少，但一般认为一旦使用现代ECC，**DRAM**软错误率非常低。 在1997年IB**M**的一份白皮书中，**De**ll [22]认为，在3年内，使用chipkill ECC的一千个一GB系统的错误率只有六个（每GB每年0.0002个错误）- 这是一个极低的错误率。 2004年T**e**zza**ro**n S**e**mic**o**nduc**tor**的一篇调查文章[83]得出结论，现代存储器设备中每**M**bi**t**的单错误率在1,000到5,000 FIT**s**（每十亿操作小时的故障次数）之间，但使用ECC可以将软错误率降低到与硬错误相当的水平。

Schroeder等人最近的一项研究[77]评估了Google服务器群体的DRAM错误，并发现FIT率远高于以前报告的水平（在25,000到75,000之间），涉及多种DIMM技术。 这意味着每年约有三分之一的Google机器会出现可纠正的内存错误，每台服务器平均每2.5小时出现一次可纠正的错误。 然而，由于ECC技术的存在，每年只有大约1.3%的机器会发生无法纠正的内存错误。

磁盘错误。基于来自N**e**tw**ork** Appl**i**anc**es** [3]、卡内基梅隆大学 [76]和G**oo**gl**e** [65]的数据，最近的研究揭示了现代磁盘驱动器的故障特性。 大型现场研究中，磁盘驱动器的硬件故障率（以更换组件的年化率衡量）通常在2%到4%之间，远高于制造商通常规定的1%或更低的数值。Ba**ir**ava**s**unda**r**am等人[3]特别研究了潜在扇区的故障率。

错误-数据损坏频率的衡量-在超过150万个驱动器的人群中。
根据他们的数据，我们可以推断出一个500GB的驱动器平均每年可能会有1.3个损坏的扇区。 这种推断有些误导，因为损坏事件并不均匀地分布在整个人群中，而是集中在一部分“坏”设备上。 例如，他们的研究发现，在32个月的时间内，不到3.5%的驱动器出现任何错误。

以上数据表明，由于磁盘或内存子系统故障而导致每年崩溃的机器的平均比例应该小于所有机器的10%。 然而，我们观察到崩溃事件更频繁，更广泛地分布在机器群体中。我们还注意到，在同质机器群体中，崩溃率存在明显的变化，这更可能是由固件和内核差异解释的。

软件引起的崩溃普遍存在的另一个间接证据是谷歌的机群中观察到的相对较长的硬件修复时间（超过6年），与机器崩溃时间（6个月或更短）相比。

重要的是要提到，良好设计的容错软件的一个关键特性是其
能够在硬件或软件错误引起的个别故障中存活。

7.3.2 预测故障

能够预测未来机器或组件故障的能力非常重要，因为这可以避免计划外停机的潜在中断。显然，能够以非常低的误报率预测给定类别故障的大多数实例的模型非常有用，特别是当这些预测涉及短期时间范围时-预测一个内存模块在未来10年内将以100%的准确率失败对于运营来说并不特别有用。

当预测准确率不完美时，这在大多数情况下很不幸是真实的，模型的成功将取决于准确性（包括误报率和时间范围）与允许故障发生和从中恢复所涉及的惩罚之间的权衡。请注意，错误的组件故障预测会产生常规硬件修复过程的所有开销（零件、技术人员时间、机器停机等）。由于WSC中的软件被设计成能够优雅地处理所有最常见的故障场景，让故障发生的惩罚相对较低；因此，预测模型必须具有更高的准确性才能在经济上具有竞争力。 相比之下，传统计算机系统中的机器崩溃可能对操作非常具有破坏性，可能会受益于准确性较低的预测模型。

Pinheiro等人[65]描述了谷歌创建基于磁盘健康参数的预测模型以预测磁盘故障的尝试。这些参数通过自我监测分析和报告技术标准获得。 他们得出结论，这样的模型不太可能预测大多数故障，并且对于模型预测的故障来说，准确性相对较低。 我们的一般经验是，只有少数故障类别可以以足够高的准确性进行预测，从而产生对WSCs有用的操作模型。

7.4 修复

高效的修复过程对WSCs的整体成本效益至关重要。处于修复状态的机器实际上无法运行，因此机器在修复中的时间越长，整体可用性就越低。 此外，修复行动在替换零件和涉及的熟练劳动力方面都是昂贵的。 最后，修复质量——修复行动实际上能否修复问题并准确确定哪个（如果有）组件有问题——会影响零件费用和平均机器可靠性。

仓库级机器有两个特点直接影响了维修效率。 首先，由于涉及大量相对低端的服务器和软件容错层的存在，快速响应个别维修案例并不是那么关键，因为它们不太可能影响整体服务健康状况。 相反，数据中心可以实施一种计划，通过每天对所有需要维修的机器进行巡检，以最高效地利用技术人员的时间。 这种理念是在保持维修延迟在可接受水平的同时提高维修速率。

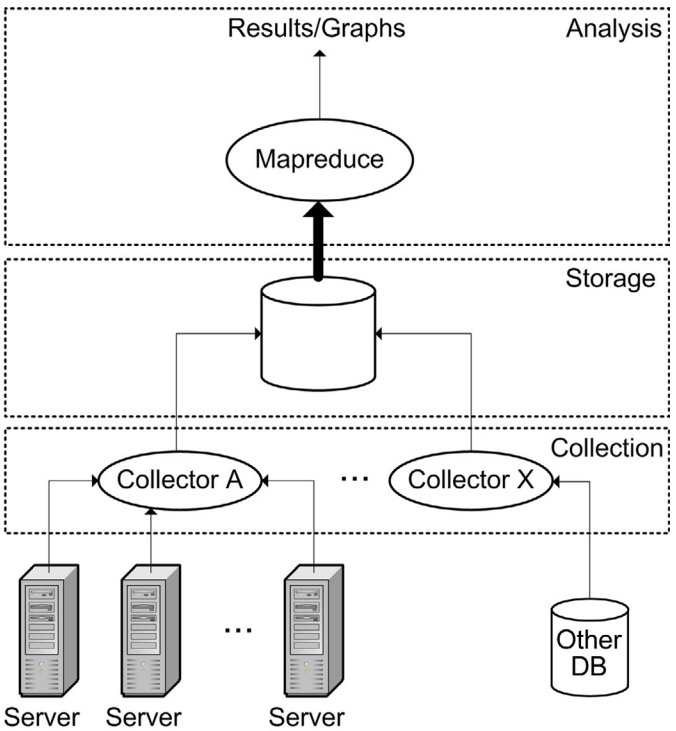

图7.4：Google系统健康监控和分析基础设施。

此外，当有成千上万台机器运行时，可以收集和分析大量关于机器健康的数据，以创建自动化的健康确定和诊断系统。 图7.4所示的Google系统健康基础设施就是利用这一海量数据源的监控系统的一个例子。 它不断监控每台服务器的配置、活动、环境和错误数据。 系统健康将这些信息作为时间序列存储在可扩展的存储库中，可以用于各种分析，包括使用机器学习方法提供最合适的维修措施的自动化机器故障诊断工具。

除了执行单个机器诊断之外，系统健康基础设施
在其他方面也很有用。 例如，它监控新系统软件版本的稳定性
并帮助确定大量机器中的特定批次的有缺陷的组件。
它还在长期的分析研究中非常有价值，例如Pinheiro等人的磁盘故障研究在前一节中提到的以及Fan等人的数据中心规模的电力供应研究[27]。

7.5 容忍故障，而不是隐藏它们

精心设计的容错软件具有掩盖大量故障的能力，对服务水平指标的影响相对较小，但可能会产生意想不到的危险副作用。 考虑一个代表具有三层结构的Web服务的应用程序，后端层被复制三次。
这样的复制设置具有增加峰值吞吐量和容忍在低于峰值容量时容忍服务器故障的双重目的。 假设传入的请求率为总容量的50%。 在这个水平上，这个设置可以在一个后端故障的情况下幸存，并且对服务水平几乎没有影响。 然而，第二个后端故障将对服务水平产生巨大影响，理论上可能导致完全停机。

这个简单的例子说明了具有大量内部冗余的系统的一个特点。 它们可以很好地容忍故障，以至于外部观察者可能不知道内部还有多少余地，或者换句话说，离边缘有多近。 在这些情况下，从正常运行到崩溃的转变可能非常突然，这是一个不可取的特性。

这个例子强调了全面监控的重要性，既在应用程序（或服务）层面上，也在机器基础设施层面上，以便故障可以被很好地容忍，同时对操作员可见。 这使得在内部冗余量接近容错软件层处理能力的极限时，能够及时采取纠正措施。

然而，损坏的机器最终必须修复。 当我们可以批量修复时，我们可以将每次修复的成本降低到传统场景中必须立即进行修复的成本以下，这需要更昂贵的备件分拣以及将服务技术人员带到现场的额外成本。 供参考，提供现场服务的IT设备服务合同

通常在24小时内修复的费用通常为设备价值的5-15%；4小时响应时间通常会使这个费用翻倍。 相比之下，大型服务器农场的维修应该更便宜。 为了说明这一点，假设一个WSC具有足够的规模来让全职维修技术人员忙碌。 假设每次维修需要1小时，年故障率为5%，一个拥有40,000台服务器的系统就足够了；实际上，这个数字会小得多，因为同一个技术人员还可以处理安装和升级。 让我们进一步假设技术人员的每小时成本为100美元，平均维修需要的替换零件费用为系统成本的10%；这两个假设都是非常慷慨的高估。 然而，对于每台成本为2,000美元的服务器集群，我们得到每台服务器的年费用为5% ∗（100美元 +10% ∗2,000美元） =15美元，或者每年0.75%。 换句话说，保持大型集群的健康状态是相当经济实惠的。

第8章

结束语

不断增长的宽带互联网接入正在使越来越多的应用程序从桌面转移到Web服务交付模式。 在这种模式下，通常称为云计算，具有大量互联处理和存储资源的数据中心可以在大量用户群体和多个普遍工作负载之间高效分摊。 这些数据中心与早期的传统协作或托管设施非常不同，构成了一类新的大规模计算机。 这些计算机中的软件由几个单独的程序构建，这些程序相互交互以实现复杂的互联网服务，并且可能由不同的工程团队设计和维护，甚至跨组织和公司边界。 这些计算机处理的数据量可以从几十到几百个TB不等，对高可用性、高吞吐量和低延迟的服务级要求通常需要基线数据集的复制。这种规模的应用程序不在单个服务器或机架上运行。 它们需要成百上千个单独服务器的集群，以及相应的存储和网络子系统、电力分配和调节设备以及冷却基础设施。

我们的核心观点很简单：这个计算平台不能简单地被视为一个杂乱无章的机器集合。 这些数据中心的大部分硬件和软件资源必须协同工作，以提供良好的互联网服务性能水平，这只能通过对其设计和部署的整体方法来实现。 换句话说，我们必须将数据中心本身视为一台巨大的计算机。 这台计算机的外壳与过去几十年中用来描述服务器的披萨盒或冰箱几乎没有任何相似之处。 相反，它更像是一座建筑物或仓库-计算机架构与传统（建筑）架构相结合。 因此，我们将这个新兴的机器类别命名为仓库级计算机（WSC）。

硬件和软件架构师需要更好地了解这类计算系统的特性，以便能够继续设计和编程今天的WSCs。但是架构师们应该记住，这些WSCs是明天日常数据中心的前身。 对于激进的多核并行性的趋势，即使是规模较小的

在几年内，计算系统将逼近今天的WSC的行为，当单个机箱中可能有数千个硬件线程可用时。

WSC由相对均匀的组件（服务器、存储和网络）构建，并在所有计算节点上使用共同的软件管理和调度基础设施来协调多个工作负载之间的资源使用。 在本节的其余部分，我们总结了前几节中描述的WSC系统的主要特点，并列出了一些重要的挑战和趋势。

8.1 硬件

WSC的首选构建模块是商品服务器级机器、消费者级或企业级磁盘驱动器和基于以太网的网络布局。 受数亿消费者和小企业的购买量推动，商品组件从制造规模经济中受益，因此其性价比明显优于相应的高端产品。 此外，互联网应用程序往往表现出大量易于利用的并行性，使得单个服务器的峰值性能不如一组服务器的总吞吐量重要。

在这个领域中，高端设备的更高可靠性并不重要，因为无论硬件质量如何，都需要一个容错软件层来提供可靠的互联网服务。在拥有数万个系统的集群中，即使是具有高可靠性服务器的集群也会频繁发生故障，以至于软件无法假设无故障运行。 此外，大型复杂的互联网服务通常由多个软件模块或层组成，这些模块或层可能比硬件组件更容易出现故障。

考虑到WSC组件的基准可靠性和典型工作负载使用的大量服务器，可能没有有用的无故障运行间隔：我们必须假设系统处于几乎连续恢复的状态。 对于需要每天每分钟保持可用的在线服务来说，这种状态尤其具有挑战性。例如，无法使用许多HPC集群常见的恢复模型，该模型在单个节点故障时会暂停整个集群工作负载，并从较早的检查点重新启动整个计算。

因此，WSC应用程序必须在软件中解决服务器故障问题，可以在应用程序级别或者（最好的情况下）通过中间件提供的功能来解决，比如一个为虚拟机提供重新启动失败虚拟机的备用节点的配置系统。 尽管低端、中度可靠的服务器构建模块对WSCs具有吸引力，但在这类系统中，高性能、高可用性的组件仍然具有价值。 例如，工作负载的一部分（如SQL数据库）可能会受益于具有更大互连带宽的高端SMP服务器。然而，高度并行的工作负载和容错软件基础设施有效地扩展了

对于WSC设计师来说，可用的构建块空间可以让较低端的选项在许多应用中表现出色。

与较小规模系统不同，网络结构和存储子系统的性能对WSC程序员来说可能更加重要，而不是CPU和DRAM子系统。 DRAM或FLASH存储的相对高成本（每千兆字节）使它们对于大数据集或不经常访问的数据来说过于昂贵；因此，仍然大量使用磁盘驱动器。 DRAM和磁盘之间性能差距的增加，以及现代磁盘驱动器吞吐量和容量之间的不平衡，使存储子系统成为大规模系统中常见的性能瓶颈。 使用许多小规模服务器需要具有非常高端口数和高双向带宽的网络结构。 由于这样的网络结构在今天是昂贵的，程序员在设计软件系统时必须非常清楚数据中心级带宽的稀缺性。 这导致了更复杂的软件解决方案，扩展的设计周期，以及有时对全局资源的低效利用。

8.2 软件

由于其规模、架构的复杂性（从程序员的角度看），以及需要容忍频繁故障，WSC（Warehouse-Scale Computers）比传统计算系统更复杂。这些传统计算系统的组件数量要小得多。

互联网服务必须实现高可用性，通常目标是99.99%或更好（每年约一小时的停机时间）。 在大规模的硬件和系统软件集合上实现无故障运行是困难的，而且由于涉及的服务器数量众多，这一任务变得更加困难。虽然在一组10,000台服务器上理论上可能防止硬件故障，但这无疑会非常昂贵。 因此，仓库规模的工作负载必须被设计成能够优雅地容忍大量组件故障，对服务级别的性能和可用性几乎没有影响。

这种工作负载与传统的高性能计算（HPC）数据中心中运行的工作负载有很大的不同，传统的HPC用户使用大规模集群计算。 与HPC应用程序一样，这些工作负载需要大量的CPU资源，但是各个任务的同步性较低，通信强度也较低。 此外，它们更加多样化，不像HPC应用程序那样只在大量节点上运行单个二进制文件。 这种工作负载中固有的并行性很大程度上是自然而易于利用的，这是因为许多用户同时访问服务或者数据挖掘中固有的并行性。 利用率会有所变化，通常会有一个日夜循环，并且很少达到90%，因为运营商更喜欢保留备用容量以应对意外的负载激增（闪电人群）或者承担其他地区失败集群的负载。 相比之下，HPC应用程序可能会连续几天或几周以全CPU利用率运行。

互联网服务的软件开发也与传统的客户端/服务器模型有许多不同之处:

- 充足的并行性—典型的互联网服务表现出大量的并行性，既来自数据并行性，也来自请求级并行性。 通常，问题不在于找到并行性，而在于管理和有效地利用应用程序中固有的显式并行性。

- 工作负载变化—互联网服务的用户通过相对明确定义和稳定的高级API（例如简单的URL）与服务的实现细节隔离，这使得快速部署新软件变得更加容易。 例如，谷歌的一些关键服务的发布周期大约为几周，而桌面软件产品则需要几个月或几年的时间。

- 平台的同质性—数据中心通常比桌面环境更具同质性。 大型互联网服务运营通常在任何给定时间点上只部署少量的硬件和系统软件配置。 显著的异质性主要来自于部署更具成本效益的组件的激励，这些组件随着时间的推移变得可用。

- 无故障运行-虽然对于桌面级软件来说，假设硬件运行数月或数年没有故障可能是合理的，但对于数据中心级服务来说，这是不正确的；互联网服务必须在故障是日常生活的一部分的环境中工作。
  理想情况下，集群级系统软件应该提供一个层，将大部分复杂性隐藏在应用级软件之下，尽管这个目标可能对于所有类型的应用来说都很难实现。

WSC硬件的复杂性作为一个编程平台可以降低编程效率，因为每个新的软件产品都必须有效地处理数据分布、故障检测和恢复，并解决性能不连续性问题（如DRAM/磁盘差距和网络结构拓扑问题）。 因此，产生隐藏这种复杂性并可以在大部分工作负载中重复使用的软件基础设施模块是至关重要的。 谷歌的MapReduce、GFS、BigTable和Chubby是WSC作为编程平台有效使用的软件的例子。

8.3 经济学

对于计算机性能和成本效益的不断需求使得成本成为设计WSC系统的主要指标。而成本效益必须

广义上定义成本效益需要考虑到成本的所有重要组成部分，包括托管设施的资本和运营费用（包括电力供应和能源成本）、硬件、软件、管理人员和维修费用。

由于其规模，WSC的功耗和能源相关成本尤为重要。
此外，固定的工程成本可以分摊到大规模部署中，高度自动化可以降低管理这些系统的成本。 因此，WSC“封闭”本身（数据中心设施、电力和冷却基础设施）的成本可能是其总成本的一个重要组成部分，因此最大限度地提高能源效率和设施利用率至关重要。 例如，智能电力供应策略，如峰值功率超额订阅，可以允许在建筑物中部署更多的系统。

WSC的利用特性使得系统和组件在广泛的负载范围内具有能效，尤其是在低利用率水平下。 使用假设操作峰值性能水平的基准测试往往高估了服务器和WSC的能效。 机器、功率转换系统和冷却基础设施在较低的活动水平下往往效率较低，例如在典型的生产系统中达到峰值利用率的30%。 我们建议将能效比例作为计算组件的设计目标之一。 理想情况下，能效比例系统在空闲时几乎不消耗电力（尤其是在活动空闲状态下），并随着活动水平的增加逐渐消耗更多电力。 能效比例的组件可以在不影响性能、可用性和复杂性的情况下大大提高WSC的能效。 不幸的是，除了CPU以外，大多数现今的组件都远未达到能效比例。

此外，数据中心本身并不特别高效。 建筑物的功耗利用效率（PUE）是总功耗与有用（服务器）功耗的比值；例如，PUE为2.0的数据中心每瓦特的服务器功耗需要1瓦特的电力。 不幸的是，许多现有设施的PUE超过2，而PUE为1.5的设施很少见。 显然，在服务器层面以及建筑层面都存在显著的提高效率的机会，正如谷歌在2008年底的自建设施中所展示的年化PUE为1.19 [32]。

能源效率优化自然会降低电力成本。 然而，供电成本，即建造一个能够提供和冷却给定功率的设施的成本，可能比电力成本本身更重要——在第6章中，我们展示了在某些部署场景中，与数据中心相关的成本可能占总IT成本的一半以上。 在管理任何大规模部署的成本中，最大化利用设施的峰值功率容量，同时降低超过该容量的风险是一个困难的问题，但非常重要。

8.4 关键挑战

我们仍在学习如何最好地设计和使用这一新型机器，但我们目前的理解使我们能够确定在这个领域中最紧迫的一些架构挑战。

8.4.1 快速变化的工作负载

互联网服务作为一个应用领域还处于起步阶段，新产品以非常快的速度出现并获得流行，其中一些服务的架构需求与它们的前身有很大不同。 例如，考虑一下YouTube视频分享网站在几个月内爆发的流行度以及这种应用的需求与之前的电子邮件或搜索等Web服务有多么不同。 WSC的一部分包括预计能够利用建筑投资超过十年的建筑结构。 激进工作负载行为变化的时间尺度与WSC的设计和生命周期之间的困难不匹配，需要硬件和软件系统的创造性解决方案。

8.4.2 从不平衡的组件构建平衡系统

尽管MHz竞赛结束，处理器仍然变得更快、更节能，因为引入了更具侵略性的多核产品。 无论是性能还是能源效率，内存系统和磁存储都没有以相同的速度发展。 这些趋势使得计算机性能和能源使用越来越多地受到非CPU组件的影响，这也适用于WSCs。最终，足够的研究重点必须转向这些其他子系统，否则处理器技术的进一步提升将不会产生明显的系统级改进。

与此同时，架构师必须努力构建高效的大规模系统，尽管可用组件存在缺陷，但在性能和成本方面显示出一定的平衡。

8.4.3 控制能源使用

随着对计算性能的渴望增加，我们必须继续寻找方法，确保性能改进伴随着能源效率的相应提高。
否则，未来计算系统的要求将占据地球有限的能源预算的越来越大份额，从而导致能源使用越来越多地限制计算能力的增长。

8.4.4 安德鲁斯残酷定律

半导体趋势表明未来的性能提升主要通过提供更多的核心或线程来实现，而不是通过更快的CPU。这意味着大规模系统必须继续提高并行效率（或加速）以处理更大、更有趣的计算问题。

系统必须继续提取更高的并行效率（或加速）来处理更大、更有趣的计算问题。

对于桌面系统来说，这是一个挑战，但对于WSC来说可能不是那么困难，因为我们之前已经提到过它在工作负载的宇宙中有丰富的线程级并行性。 话虽如此，即使是高度并行的系统也遵循阿姆达尔定律，可能会有一个点，在这个领域中阿姆达尔效应变得主导。这一点可能会提前出现；例如，如果高带宽、高端口数量的网络技术相对于其他WSC组件仍然非常昂贵的话。

8.5 结论

计算正在转移到云端，因此也转移到了WSC。软件和硬件架构师必须了解端到端系统，以设计出好的解决方案。 我们不再设计单独的“披萨盒”，或者单服务器应用程序，也不能再忽视仓库里的计算机所涉及的物理和经济机制。 从某种程度上说，WSC很简单——只是通过局域网连接的几千台廉价服务器。实际上，构建一个具有必要可靠性和可编程性要求的、成本高效的大规模计算平台，以满足下一代云计算工作负载的需求，与计算机系统中的任何其他挑战一样困难且激动人心。 我们希望这本书能帮助计算机科学家们理解这个相对较新的领域，并且相信在未来的几年里，他们的共同努力将解决许多与仓库规模系统相关的有趣问题。

参考文献

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, “可扩展的、通用的数据中心网络架构”，2008年ACM SIGCOMM会议论文集，西雅图，华盛顿州，2008年8月17日至22日。

[2] D. Atwood and J. G. Miner, “通过空气经济装置降低数据中心成本”，Intel IT简报，2008年8月。

[3] L. N. Bairavasundaram, G. R. Goodson, S. Pasupathy, and J. Schindler, “磁盘驱动器中潜在扇区错误的分析”，2007年ACM SIGMETRICS国际会议论文集，圣地亚哥，加利福尼亚州，2007年6月12日至16日。SIGMETRICS '07。 doi:10.1145/1254882.1254917

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, “Xen和虚拟化的艺术,” in Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, Bolton Landing, NY, October 19–22, 2003. SOSP ’03. doi:10.1145/945445.945462

[5] L. Barroso, J. Dean, and U. Hölzle, “Web搜索一个星球：Google集群的架构,” IEEE Micro, April 2003. doi:10.1109/MM.2003.1196112

[6] L. A. Barroso and U. Hölzle, “能源比例计算的案例,” IEEE Computer, vol. 40, no. 12 (Dec. 2007).

[7] M. Burrows，“松散耦合分布式系统的肥胖锁服务”，在OSDI’06:第七届操作系统设计与实现研讨会上，西雅图，华盛顿，2006年11月。

[8] P. Bahl，R. Chandra，A. Greenberg，S. Kandula，D. A. Maltz和M. Zhang，“通过推断多级依赖关系实现高可靠性企业网络服务”，在SIGCOMM会议论文集中，2007年。

[9] P. Barham，R. Isaacs，R. Mortier和D. Narayanan，“Magpie:在线建模和性能感知系统”，在USENIX HotOS IX 2003会议论文集中。

[10] E. A. Brewer，“巨型规模服务的经验教训”，IEEE互联网计算，第5卷，第4期（7月/8月2001年），第46-55页。doi:10.1109/4236.939450

[11] E. V. Carrera, E. Pinheiro, and R. Bianchini, “在网络服务器中节约磁盘能量”，在第17届年度国际超级计算机会议上的论文集，旧金山，加利福尼亚，2003年6月23日至26日。ICS '03。 doi:10.1145/782814.782829

[12] B. Chandra, M. Dahlin, L. Gao, and A Nayate, “端到端的广域网服务可用性”，在第3届USENIX互联网技术和系统研讨会论文集，旧金山，加利福尼亚，2001年3月26日至28日。

[13] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable: 用于结构化数据的分布式存储系统”，在OSDI 2006论文集中，西雅图，华盛顿州，2004年。

[14] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle, “在托管中心管理能源和服务器资源,” in Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), Banff, Alberta, Canada, June 2001. doi:10.1145/502034.502045[15] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, “面向连接密集型互联网服务的能源感知服务器供应和负载调度,” Microsoft Research Technical Report MSR-TR-2007-130, 2007.

[16] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, “在托管中心管理服务器能源和运营成本,” in Proceedings of the ACM SIG-METRICS ’05, Banff, Alberta, Canada, June 2005. doi:10.1145/1064212.1064253[17] Climate savers computing efficiency specs. 可在http://www.climatesaverscomputing.org/about/tech-specs找到.

[18] E. F. Coyle，“改进的肌肉效率显示为环法自行车冠军成熟”，应用生理学杂志，2005年3月。 doi:10.1152/japplphysiol.00216.2005[19] J. Dean和S. Ghemawat，“MapReduce：简化大型集群上的数据处理”，ACM通信，第51卷，第1期（2008年），107-113页。

[20] G. DeCandia，D. Hastorun，M. Jampani，G. Kakulapati，A. Lakshman，A. Pilchin，S. Siv-asubramanian，P. Vosshall和W. Vogels，“Dynamo：亚马逊高可用性键值存储”，在第21届ACM操作系统原理研讨会上的论文集，华盛顿州斯蒂文森，2007年10月。

[21] Dell能源计算器。可在http://www.dell.com/calc上获得。

[22] T. J. Dell，“关于PC服务器主内存中Chipkill-correct ECC的好处的白皮书”，IBM微电子部门，Rev 11/19/1997。

[23] D. Dyer，“数据中心热管理的当前趋势/挑战——设施角度的观点”，2006年6月1日在ITHERM上的演讲，圣地亚哥，加利福尼亚州。

[24] J. Elerath和S. Shah，“服务器级磁盘驱动器：它们可靠吗？”，IEEE可靠性和可维护性，2004年年度研讨会—RAMS，2004年1月。 doi:10.1109/RAMS.2004.1285439

[25] 杜邦法布罗斯技术公司SEC文件（S-11）333-145294，2007年8月9日。

[26] 美国环境保护局，“关于服务器和数据中心能效的报告”，公共法案109-431，2007年8月2日。

[27] X. Fan, W. Weber和L. A. Barroso，“用于仓库规模计算机的电源配置”，在第34届国际计算机体系结构研讨会论文集中，圣地亚哥，加利福尼亚，2007年6月9日至13日。ISCA '07。 doi:10.1145/1250662.1250665[28] M. E. Femal和V. W. Freeh，“安全过度配置：使用功率限制来增加总吞吐量”，在第4届国际节能计算机系统研讨会论文集中，2004年12月，第150-164页。

[29] M. E. Femal和V. W. Freeh，“通过非均匀功率分配提高数据中心性能”，在第二届国际自动计算会议论文集中，2005年6月13日至16日，第250-261页。

[30] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, 和 I. Stoica, “X-trace: 一个普遍的网络追踪框架,” 在第4届USENIX网络系统设计与实现研讨会论文集, 2007, pp. 271–284.

[31] S. Ghemawat, H. Gobioff, 和 S-T. Leung, “Google文件系统”, 在第19届ACM操作系统原理研讨会论文集, Lake George, NY, 2003年10月. doi:10.1145/945445.945450

[32] Google公司, “高效计算—第2步: 高效数据中心”. 网址为 http://www.google.com/corporate/green/datacenters/step2.html.

[33] Google公司, “高效数据中心峰会, 2009年4月”. 网址为 http://www.google.com/corporate/green/datacenters/summit.html.

[34] 谷歌公司，“高效的数据中心，第一部分”。 可在http://www.youtube.com/watch?v=Ho1GEyftpmQ上观看，从0:5930开始。

[35] J. Gray，“1985年至1990年间串联系统可用性普查”，Tandem技术报告90.1，1990年1月。

[36] The Green 500。 可在http://www.green500.org上获取。

[37] The Green Grid数据中心功耗效率指标：PUE和DCiE。可在http://www.thegreengrid.org/sitecore/content/Global/Content/white-papers/The-Green-Grid-Data-Center-Power-Efficiency-Metrics-PUE-and-DCiE.aspx上获取。

[38] Green Grid，“数据中心电力分配配置的定量分析”。可在http://www.thegreengrid.org/gg_content/上获取。

[39] 绿色网格，“提高数据中心冷却效率的七种策略”。 可在http://www.thegreengrid.org/gg_content/上找到。

[40] SNIA绿色存储倡议。可在http://www.snia.org/forums/green/上找到。[41] S. Greenberg, E. Mills和B. Tschudi，“数据中心的最佳实践：从22个数据中心的基准测试中学到的经验”，2006年ACEEE夏季研讨会上的能源效率研究。 可在http://eetd.lbl.gov/EA/mills/emills/PUBS/PDF/ACEEE-datacenters.pdf上找到。

[42] Hadoop项目。可在http://hadoop.apache.org上找到。

[43] J. Hamilton，“关于设计和部署互联网规模服务”，在USENIX LISA会议论文集中，2007年。

[44] J. Hamilton，“合作式可替换微片服务器（CEMS）：互联网规模服务的低成本、低功耗服务器”，在第四届创新数据系统研究双年会议（CIDR）上，加利福尼亚州阿西洛马尔，2009年1月4日至7日。

[45] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, 和 R. Bianchini, “Mercury and freon: temperature emulation and management for server systems,” in Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, October 2006.

[46] HP Power Calculator. 可在 http://www.hp.com/configurator/powercalcs.asp.

找到。[47] M. Isard, M. Budiu, Y. Yu, A. Birrell, 和 Fetterly, “Dryad: distributed data-parallel programs from sequential building blocks,” in Proceedings of the 2nd ACM Sigops/Eurosys European Conference on Computer Systems 2007, Lisbon, Portugal, March 21–23, 2007.

[48] M. Isard, “Autopilot: automatic datacenter management”. SIGOPS Operating Systems Review, vol. 41, no. 2 (Apr. 2007), pp. 60–67. DOI: http://doi.acm.org/10.1145/1243418.1243426.

[49] M. Kalyanakrishnam, Z. Kalbarczyk, 和 R. Iyer, “Failure data analysis of a LAN of Windows NT based computers,” Reliable Distributed Systems, IEEE Symposium on, vol. 0, no. 0, pp. 178, 18th IEEE Symposium on Reliable Distributed Systems, 1999. doi:10.1109/RELDIS.1999.805094

[50] J. Koomey, K. Brill, P. Turner, J. Stanley and B. Taylor, “一个用于确定数据中心真正总拥有成本的简单模型”，Uptime Institute白皮书，版本2，2007年10月。 doi:10.1145/1394608.1382148

[51] K. Lim, P. Ranganathan, J. Chang, C. Patel, T. Mudge, and S. Reinhardt, “理解和设计新的服务器架构以适应新兴的仓库计算环境”，计算机体系结构，国际研讨会，卷0，号0，页315-326，2008年国际计算机体系结构研讨会，2008年。

[52] C. Malone and C. Belady, “用于表征数据中心和IT设备能源使用的度量标准”，在数字电源论坛会议论文集中，理查森，德克萨斯州，2006年9月。

[53] 迈克·马诺斯。“容量问题”。可在http://loosebolts.wordpress.com/2009/06/02/chiller-side-chats-the-capacity-problem/上找到。

[54] W. D. McArdle, F. I. Katch和V. L. Katch，Sports and Exercise Nutrition，第二版，LWW Publishers，2004年。

[55] D. Meisner，B. Gold和T. Wenisch，“PowerNap：消除服务器空闲功耗”，在Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems，华盛顿特区，2009年3月。

[56] J. Moore，J. Chase，P. Ranganathan和R. Sharma，“使调度‘酷’：数据中心中的温度感知工作负载放置”，在年度USENIX技术会议上的会议论文集，加利福尼亚州阿纳海姆，2005年4月10日至15日。

[57] D. Nelson, M. Ryan, S. DeVito, K. V. Ramesh, P. Vlasaty, B. Rucker, B. Da y Nelson, et al., “模块化在数据中心设计中的作用”。 Sun BluePrints Online, http://www.sun.com/storagetek/docs/EED.pdf。

[58] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, “为什么互联网服务会失败，以及如何解决？”，在第四届USENIX互联网技术和系统研讨会论文集第4卷，西雅图，华盛顿州，2003年3月26日至28日。

[59] L. Page and S. Brin, “一个大规模超文本搜索引擎的解剖”。 可在http://infolab.stanford.edu/~backrub/google.html找到。

[60] C. Patel等，“冷却大规模高计算密度数据中心的热量考虑”。 可在http://www.flomerics.com/flotherm/technical_papers/t299.pdf找到。[61] C. Patel等，“数据中心规划、开发和运营的成本模型”。 可在http://www.hpl.hp.com/techreports/2005/HPL-2005-107R1.pdf找到。[62] D. A. Patterson, A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A.Fox, E. Kiciman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupman,and N. Treuhaft.“面向恢复的计算（ROC）：动机、定义、技术和案例研究”。 UC Berkeley计算机科学技术报告UCB//CSD-02-1175，2002年3月15日。

[63] M. K. Patterson和D. Fenwick，“数据中心冷却的现状”，英特尔公司白皮书。可在http://download.intel.com/technology/eep/data-center-efficiency/state-of-date-center-cooling.pdf获取。

[64] R. Pike，S. Dorward，R. Griesemer和S. Quinlan，“解释数据：使用Sawzall进行并行分析”，科学编程杂志，第13卷，第4期（2005年），第227-298页。

[65] E. Pinheiro，W.-D. Weber和L. A. Barroso，“大型磁盘驱动器群体的故障趋势”，第5届USENIX文件和存储技术会议（FAST2007）论文集，加利福尼亚州圣何塞，2007年2月。

[66] PG&E，“高性能数据中心”。 可在http://hightech.lbl.gov/documents/DATA_CENTERS/06_DataCenters-PGE.pdf获取。

[67] W. Pitt Turner IV, J. H. Seader, and K. G. Brill, “层级分类定义站点基础设施性能”，Uptime Institute白皮书。

[68] W. Pitt Turner IV和J. H. Seader，“每千瓦美元加上每平方英尺美元是一个更好的数据中心成本模型，而不仅仅是每平方英尺美元”，Uptime Institute白皮书，2006年。

[69] R. Raghavendra，P. Ranganathan，V. Talwar，Z. Wang和X. Zhu，“没有‘功率’斗争：数据中心的协调多级功率管理”，ACM会议论文集

国际支持编程语言和操作系统的体系结构支持会议，西雅图，华盛顿州，2008年3月。

[70] 路透社，“Dupont Fabros Technology, Inc.报告第三季度业绩”，2008年11月5日。

[71] P.雷诺兹，C.基利安，J.L.维纳，J.C.莫格尔，M.A.沙阿和A.瓦达特，“Pip:检测分布式系统中的意外情况”，在USENIX NSDI会议论文集中，2006年。

[72] P.雷诺兹，J.L.维纳，J.C.莫格尔，M.K.阿吉莱拉和A.瓦达特，“WAP5:用于广域系统的黑盒性能调试”，在第15届国际万维网会议论文集中，2006年。

[73] 罗伯特·弗朗西斯集团，“企业中Linux的总拥有成本”，2002年7月。 可在http://www-03.ibm.com/linux/RFG-LinuxTCO-vFINAL-Jul2002.pdf获取。doi:10.1088/1742-6596/78/1/012022

[74] SAVVIS新闻稿，“SAVVIS以2亿美元出售与两个数据中心相关的资产”，2007年6月29日。 可在http://www.savvis.net/corp/News/Press+Releases/Archive/SAVVIS+Sells+Assets+Related+to+Two+Data+Centers+for+200+Million.htm获取。

[75] B. Schroeder和G. A. Gibson，“理解拍它尺度计算机的故障”，《物理学杂志：会议系列78》（2007年）。

[76] B. Schroeder和G. A. Gibson，“现实世界中的磁盘故障：100万小时的MTTF对你意味着什么？”，在《第5届USENIX文件和存储技术会议论文集》中，2007年2月。

[77] B. Schroeder，E. Pinheiro和W.-D. Weber，“野外DRAM错误：大规模实地研究”，将出现在SIGMETRICS的论文集中，2009年。

[78] S. Siddha，V. Pallipadi和A. Van De Ven，“充分利用无滴答的最大里程”，在《Linux研讨会论文集》中，加拿大安大略省渥太华，2007年6月。

[79] SPEC Power。可在http://www.spec.org/power_ssj2008/上获得。

[80] S. Rivoire, M. A. Shah, P. Ranganathan, C. Kozyrakis, and Justin Meza, “用于能效优化的模型和度量标准”，计算机，第40卷，第12期（2007年12月），第39-48页。

[81] S. Sankar, S. Gurumurthi, and M. R. Stan, “硬盘内部并行性：一个时机已到的想法”，ACM国际计算机体系结构研讨会论文集，2008年6月，第303-314页。

[82] Techarp.com，“Intel桌面CPU指南”。 可在http://www.techarp.com/showarticle.aspx?artno=337&pgno=6找到。

[83] Terrazon Semiconductor，“电子存储器中的软错误——白皮书”。 可在http://www.tezzaron.com/about/papers/soft_errors_1_1_secure.pdf找到。

[84] M. Ton和B. Fortenbury，“高性能建筑：数据中心-服务器电源供应”，劳伦斯伯克利国家实验室和EPRI，2005年12月。

[85] 事务处理性能委员会。可在http://www.tpc.org获得。[86] W. F. Tschudi，T. T. Xu，D. A. Sartor和J. Stein，“高性能数据中心：研究路线图”，劳伦斯伯克利国家实验室，加利福尼亚州伯克利，2003年。

[87] TPC-C超级穹顶-Itanium2的执行摘要，2007年2月。

[88] TPC-C ProLiant ML350G5的执行摘要，2007年9月。

[89] VMware基础架构架构概述白皮书。 可在http://www.vmware.com/pdf/vi_architecture_wp.pdf获得。

[90] W. Vogels，“最终一致性”，ACM Queue，2008年10月。 可在http://queue.acm.org/detail.cfm?id=1466448获得。

[91] B. Chandra, M. Dahlin, L. Gao, A.-A. Khoja, A .Nayate, A. Razzaq, and A. Sewani, “可扩展断开式访问Web服务的资源管理”，第10届国际万维网会议，香港，中国，2001年5月1日至5日，第245-256页。

[92] “微软可持续发展数据中心的前10项业务实践。” 可在http://www.microsoft.com/environment/our_commitment/articles/datacenter_bp.aspx上找到。

作者简介

Luiz André Barroso是Google的杰出工程师，在多个工程领域工作过，包括软件基础设施、故障分析、能源效率和硬件设计。 **Luiz**还是Google平台工程团队的首任经理，该团队负责设计公司的计算平台。 在加入Google之前，他曾是Digital Equipment Corporation（后来被康柏收购）的研究人员，他的团队在商业工作负载的处理器和内存系统设计方面做出了一些开创性的工作。

这项研究导致了Piranha的设计，这是一种单芯片多处理器，它启发了现在主流的一些多核CPU。 在加入Digital之前，Luiz是USC RPM的设计师之一，这是一种基于可编程门阵列的多处理器仿真器，用于快速硬件原型设计。 他曾在巴西里约热内卢的Pontifícia Universidade Católica (PUC)-Rio和斯坦福大学讲课，拥有南加州大学的计算机工程博士学位和PUC里约热内卢的电气工程学士/硕士学位。

Urs Hölzle担任谷歌首位工程副总裁，并领导了谷歌技术基础设施的开发。 他目前的职责包括设计和运营支撑谷歌的服务器、网络和数据中心。他以他的红袜子和谷歌的顶级狗狗Yoshka而闻名。 **Urs**从加州大学圣塔芭芭拉分校加入谷歌，他曾任计算机科学副教授。 他于1988年从苏黎世联邦理工学院获得计算机科学硕士学位，并在同年获得富布赖特奖学金。 1994年，他在斯坦福大学获得博士学位，他的研究重点是编程语言及其高效实现。

作为动态编译的先驱之一，也被称为“即时编译”，Urs发明了大多数当今领先的Java编译器中使用的基本技术。 在加入Google之前，Urs是Animorphic Systems的联合创始人，该公司开发了Smalltalk和Java的编译器。1997年Sun Microsystems收购Animorphic Systems后，他帮助构建了Javasoft的高性能Hotspot Java编译器。