分　数：	
评卷人：	

華中科技大學

研究生（数据中心技术）课程论文（报告）

题 目： 数据中心电力优化方法综述

学　　号 M202173487

姓　　名 廖子逸

专　　业 电子信息

课程指导教师 施展 童薇

院（系、所） 武汉光电国家研究中心

2021 年 1 月 3 日

数据中心电力优化方法综述

廖子逸[1)]

[1)]（华中科技大学武汉光电国家研究中心 武汉 430000）

摘 要 云提供商，如亚马逊和微软，必须保证云服务上工作负载的高可用性。因此，他们构建的数据中心包含大量冗余基础设施，用于电力输送等方面。通常，冗余资源将在基础设施故障期间起到作用，来保证工作负载性能和可用性不受影响，同时也说明冗余资源的利用率低。为了提高资源利用率，数据中心常用的做法是结合超额预购（oversubscription）和功率封顶（power capping）来让服务器在保证安全的情况下获得更多电力。但传统的方法比较保守，仍然有较多搁浅电力（stranded power）被浪费。基于以上情况，文章 1：A Scalable Priority-Aware Approach to Managing Data Center Server Power(HPCA 19)提出了 CapMaestro：一种基于可扩展优先级感知方法的数据中心服务器电源管理方法；文章 2：Prediction-Based Power Oversubscription in Cloud Platforms(USENIX ATC 21)提出了一种利用机器学习方法预测虚拟机性能和利用率从而优化超额预算和功率封顶的方法；文章 3：Flex: High-Availability Datacenters With Zero Reserved Power(ISCA 21)提出了 Flex：零预留电力的高可用性数据中心电源管理方法。在实验部分，文章 1 证明了 CapMaestro 显着且安全地增加了现有数据中心基础设施中可部署服务器数量；文章 2 的方法相对现有先进方法提高了两倍的超额预算值；文章 3 的 Flex 系统将电力搁浅率降低至 5%以下，且相较传统方法提升了 33%的服务器部署数量。

关键词 数据中心；超额预购；功率封顶；预留电力；搁浅功率

A survey of Data Center Power Optimization Methods

Ziyi Liao[1)]

[1)]（Wuhan National Laboratory for Optoelectronics，Huazhong University of Science and Technology Wuhan 430000）

Abstract Cloud providers, such as Amazon and Microsoft, must ensure high availability of workloads on cloud services. Therefore, the data center they built contains a large amount of redundant infrastructure for power transmission and other aspects. Generally, redundant resources will play a role during infrastructure failures to ensure that workload performance and availability are not affected, and it also indicates that the utilization of redundant resources is low. In order to improve resource utilization, a common practice in data centers is to combine oversubscription and power capping to allow servers to obtain more power while ensuring safety. However, the traditional method is conservative, and more stranded power is still wasted. Based on the above situation, Article 1: A Scalable Priority-Aware Approach to Managing Data Center Server Power (HPCA 19) proposed CapMaestro: A Data Center Server Power Management Method Based on Scalable Priority Awareness; Article 2: Prediction-Based Power Oversubscription in Cloud Platforms (USENIX ATC 21) proposed a method of using machine learning methods to predict the performance and utilization of virtual machines to optimize excess budget and power capping; Article 3: Flex: High-Availability Datacenters With Zero Reserved Power (ISCA 21) Proposed Flex: a high-availability data center power management method with zero reserved power. In the experimental part, article 1 proves that CapMaestro significantly and safely increases the number of servers that can be deployed in the existing data center infrastructure; the method of article 2 increases the excess budget by two times compared with the existing advanced methods; the Flex of article 3 The system reduces the power stranding rate to less than 5% and increases the number

of server deployments by 33% compared to traditional methods.

Key words data center; oversubscription; power capping; reserved power; stranded power

1 引言

随着对云服务（例如，基础设施即服务，Infrastructure as a Service, IaaS）的需求不断增长，云提供商必须建立更多高可用性数据中心。供应商为了减少基础设施建设成本，减少需要建设的数据中心数量，必须高度利用每个数据中心的基础资源（例如，电力、空间、冷却）。由于电力通常是瓶颈资源，大型提供商使用电力超额预购和功率封顶（性能限制）实现向每个数据中心部署更多服务器来安全地提高利用率。传统方法下，提供商保守地选择电力超额预购比例，以防止因功率限制而过多影响性能。

同时，因为数据中心高可用性的需求，提供商在数据中心部署大量冗余资源，包括电力和冷却，以确保在计划内维护或基础设施故障（也称为计划外维护）的情况下数据中心的可用性。超额预购和功率封顶只会限制系统峰值功率，从而忽略了预留部分的资源。也就是说，只有当系统出现故障或者功率超额的情况下，这些预留电力才会被使用，而在正常运行期间，预留电力往往被浪费了。充分利用这部分预留电力来部署服务器可以提高数据中心利用率，降低数据中心基础设施成本。

文章3根据微软数据中心实际部署经验，总结了三个重要观察结论：

（1） 云服务通常表现出相对较低的电力利用率，偶尔会出现峰值。因此，即使采用最先进的超额订阅策略，关键基础设施故障（或计划维护）和电力峰值同时发生的概率也很小。

（2） 提供商通常在每个数据中心托管具有不同基础架构可用性要求的混合工作负载，其中的一些软件冗余工作负载（例如，Web 搜索、数据分析）可以由其软件内置冗余而容忍基础架构故障。

（3） 非固有冗余的工作负载，（例如，单个 IaaS 虚拟机）不能容忍基础架构故障。但是，其中一些（例如，first-party VMs，Virtual machines）通常可以容忍轻微的性能限制。此类工作负载的存在使得通过节流将功率超额订阅和电源上限相结合成为可能。

观察结论（1）表明维护事件很少需要采取纠正措施来降低功耗，因为使用保留资源配置的服务器的电源利用率也只是偶尔出现峰值。观察（2）和（3）表明在故障期间或者电力峰值超过额定功率时可以关闭一些软件冗余工作负载所在的机架，或者限制一些可以容忍轻微性能限制负载的机架的性能来降低功率。这让通过更大胆的超额预购实现更高的电力利用率成为可能。

基于以上分析，文章 1[1]（A Scalable Priority-Aware Approach to Managing Data Center Server Power）和文章 3[3]（Flex: High-Availability Datacenters With Zero Reserved Power）从数据中心服务提供者的角度进行电力控制优化。通过文章 1 和文章 3 的验证，这种超额预购方式在负载类型、部署位置以及服务器物理特性已知的情况下表现良好。但公有云平台类似黑盒，内部结构不清晰。具体来说：（1）每个服务器上运行着性能、电力需求不同的 VMs，机架级别电力限制可能会影响重要 VMs 的性能（与付费用户交互的 VMs）。（2）VMs 动态的到达和离开服务器。（3）云平台通常不具有访问 VMs 内部运行情况的权限。由于这些原因，公共云中的超额订阅受到限制，来保证重要VMs的性能永远不会受到影响。基于此原因，文章 2[2]（Prediction-based power oversubscription in cloud platforms）提出了基于机器学习的 VMs 性能预测方法，从服务本身的角度进行电力控制优化。

2 原理和优势

2.1 A Scalable Priority-Aware Approach to Managing Data Center Server Power

文章 1 总结了（文章 1 发布于 19 年）现有的基于功率上限的方法无法减少配电基础设施中的大量过度配置，主要原因有三个：

（1）在高可用性数据中心里，通常配置了多源电力，每个服务器都连着多个电力供应设备来分担电力负载。而在一台服务器上，对于不同电力供应设备的电力负载分配往往是不平衡的，这将导致有的电力供应设备过载而降低功率上限，但其实整个数据中心总功率并未超标。

（2）现有的功率限制方法没有考虑负载优先级，或者只考虑小部分的本地服务器群优先级。

（3）服务器上的负载通常大小不定，而且到达时间和完成时间随机，这将造成电力分配过程中出现“碎片”电力，即搁浅电力。

基于以上三点问题，文章 1 提出了 CapMaestro，一种新的公有云数据中心电源管理架构。相比之前的工作，CapMaestro 有以下三个特点：

（1）CapMaestro 使用一种新的闭环反馈电源控制器来保证电源架构中各级供电设备不会过载。

（2）CapMaestro 通过一个高效的分布式全局优先级感知算法实现跨多个协调功率控制器分配电力，实现更高的容错能力，提高了超额预算比例。这使得 CapMaestro 可以从全局角度调配电力，在故障期间保证低优先级负载最低电力需求的同时将更多的电力调配给高优先级负载。

（3）CapMaestro 通过调整服务器电源预算优化服务器的搁浅电力，提高电力利用率。

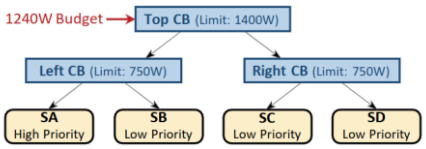


图 1 具有多优先级的多源电力供应例子

如图 1 所示是一个具有多优先级的多源电力供应例子。其中 CB（Circuit Breaker）是电力结构保护设备，用于防止级联故障。CB 的值是该设备的最高负载值。在图 1 中，四个服务器 SA,SB,SC 和 SD 连接到三个 CB 的供电电源：一个额定功率为 1400W 的顶级 CB，以及两个额定功率为 750W 的左 CB 和右 CB。 这四个每个服务器的功率需求相等，为 430W，每个服务器的最小运行功率 Pcap_min = 270W，其中 SA 为高优先级服务器。

表 1 全局优先级和本地优先级下的电源预算对比

Server	SA	SB	SC	SD
Priority (H = high, L = low)	H	L	L	L
Power Demand (W)	430	430	430	430
Budget with Local Priority (W)	350	270	310	310
Budget with Global Priority (W)	430	270	270	270

表 1 的第 3、4 行说明了在全局优先级电源预算下，当总功率受到限制时候，SA 在基于全局优先级下可以调配到另一个上层 CB（Right CB）下的电力预算，能获得更多的电源预算（430W），而基于本地优先级电源预算下，SA 只能调配本地 CB（Left CB）的功率预算，获得更少的电源预算（350W）。

2.2 Prediction-based power oversubscription in cloud platforms

文章 2 使用基于机器学习的预测方法来预测 VM 重要性和资源利用率，从而优化超额预算策略。文章 2 提出的方法具体分为预测 VM 重要性和预测 VM 资源利用率两个部分。

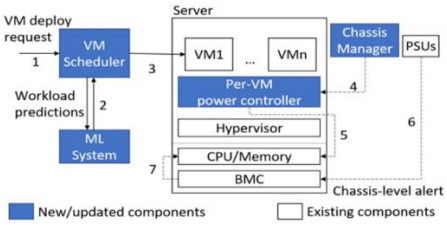


图 2 文章 2 中的系统架构

图 2 概述了文章 2 中的系统及其操作，其中不同蓝色的方框代表新组件和旧组件。文章 2 在原有系统上加入一个 VM 调度器，调度器使用了基于机器学习的 VM 特性预测结果来帮助更好的在服务器中部署 VMs。同时，文章 2 也修改了底层电源管理器，管理器通过一个新加入的 VM 电源控制器来保证整个系统安全运行。结合图 2 中标注的数字，详细讲解系统运行过程：

（1）一组 VM 部署请求输入到 VM 调度器。

（2）调度器查询 ML 系统获得 VM 的关键性和利用率预测。

（3）调度器使用预测值来决定 VM 的实际部署。

（4）底层电源管理器会在系统功率超标的情况下对机箱中每个服务器的控制器发出警报。

（5）收到警报后，每个服务器的控制器通过限制非重要 VM 所在服务器的 CPU 功率来降低电力。

（6）若操作 5 无法将系统功率降低至安全线以下，PSUs（Power Supplies）会将信息发给底层 BMCs（Baseboard Manager Controllers）。

（7）BMCs 会使用运行时平均功率限制(Runtime Average Power Limit，RAPL)强制降低服务器功率。

上述系统通过对数据中心负载（VMs）的分析与预测，能够实现更好的超额预算策略，提高资源利用率，降低数据中心构建成本。

2.3 Flex: High-Availability Datacenters With Zero Reserved Power

文章 3 提出的 Flex 系统旨在优化超额订阅策略，充分利用搁浅电力来部署更多服务器，同时将峰值功率消耗保持在安全范围内。先前的工作中，文章 1 提出的 CapMaestro 是首个使用预留电源部署更多服务器的系统。但是，CapMaestro 不会在工作负载放置或故障转移期间考虑每个工作负载的可用性要求。这限制了可以使用的保留功率量。如图 3 所示，上半部分是传统的超额预算策略，下半部分是 Flex 的实现零预留电力超额预算策略。相比之下，Flex 可以将全部预留电源用于服务器部署，并满足每个工作负载所需的可用性需求。

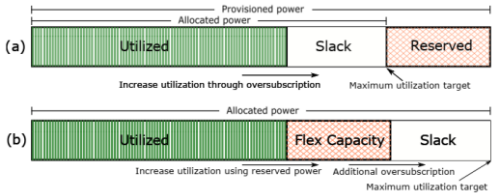


图 3 零预留电力概念示意图

实现 Flex 系统有三个难点：

（1）当数据中心出现故障的时候，故障服务器上的负载必须快速转移到冗余设备上，并保证各级供电设备不过载。

（2）工作负载必须放置在当前数据中心，以保证安全，避免故障，同时最大限度地减少搁浅电力。

（3）在维护事件期间，数据中心必须在可以持续透支的短时间（UPS 容差时间）内采取纠正措施，否则会产生整个数据中心断电的事故。由于基础设施没有有关工作负载特征和要求的信息，因此纠正策略的选择必须在后台软件中执行，但数据链路存在潜在的服务器和通信故障。

为了应对这些挑战，微软构建了 Flex 系统，主要包括以下三个部分：

（1）Flex 系统扩展了最新一代 Microsoft 数据中心的高可用性电源基础结构，以确保它能够获取更多电源并留出足够的时间在软件中采取纠正措施。

（2）Flex 系统引入了一种新的离线工作负载放置策略（Flex-Offline），可减少闲置电源，同时确保维护事件期间的安全。Flex-Offline 将负载放置建模为整数线性规划问题，通过优化负载放置提高资源利用率和安全性。

（3）Flex 系统构建了一个高度可用的

分布式系统（Flex-Online），动态监控基础设施故障并采取纠正措施，在几秒钟内将电源降至故障转移预算以下。Flex-Online 由两部分组成：通过高度冗余设施实现的高精度、高可用性电力遥测系统和动态管理电源保证服务器可用性和性能的运行决策系统。

实验表明，Flex 系统管理下的微软数据中心中搁浅电力占比小于 4%，比简单策略减少了 37%。Flex 可以在保证安全，满足不同负载需求的情况下将每个数据中心的服务器部署数量增加 33%。

3 研究进展

3.1 数据中心电力结构

为了提供高可用性基础架构，云提供商使用冗余电源设备和线路构建数据中心。例如配置冗余的不间断电源 (Uninterruptible Power Supplies, UPSes)、发电机、变压器和配电单元 (Power Distribution Units, PDU)等。冗余形式有多种，比如单备份（N+1）、双备份（2N）、和分布式备份（xN/y），每种形式在成本、可维护性和容错性方面有自己的权衡。基于这些权衡，设计师经常在数据中心中使用不同的电源层次结构中不同级别的冗余模式。

3.1.1 2N 冗余度电力结构

如图 4 所示是文章 1 中的数据中心高可用性电力结构。该结构冗余度为 2N，即有两侧完全一样的供电设备对一组服务器进行供电，当一侧供电设备发生故障时候，负载全部转移到另一侧供电设备中。所以在 2N 冗余度电力结构中，最保守的做法是每一侧的功率不超过供电设备额定功率的一半，这也将导致预留电力占比为 50%。如图 4 所示，CB（Circuit Breaker）处于④处，用于在系统过载时保护设备而强制断电。通常 CB 有一个容许时间（约 30s），代表着电力控制器需要在 CB 的容许时间内完成故障时期负载转移以及后续降低功率的动作。

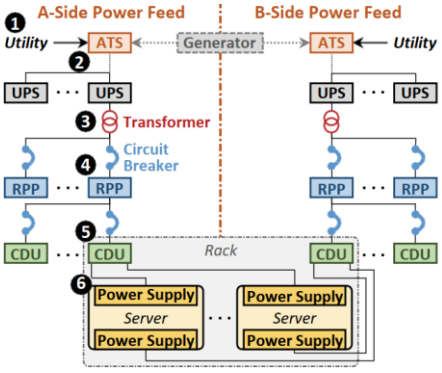


图　4 2N 冗余度电力设施架构

3.1.2 4N/3 冗余度电力结构

如图 5 所示是文章 3 中的数据中心高可用性电力结构。该结构冗余度为 4N/3，即有四组完全一样的供电设备对三组服务器进行供电，当一组供电设备发生故障时候，负载全部转移到另外三组供电设备中。所以在 4N/3 冗余度电力结构中，最保守的做法是每组供电设备的功率不超过供电设备额定功率的 3/4，这也将导致预留电力占比为 25%，显然少于 2N 冗余度电力结构，但同时也提高了系统复杂性和构建成本。

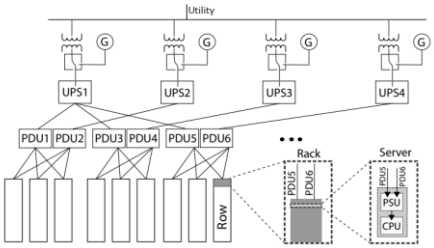


图　5 4N/3 冗余度电力设施架构

3.2 数据中心负载和硬件特点

大型云提供商提供多种服务，比如提供 VMs 的架构即服务（IaaS），提供终端用户服务的软件即服务（SaaS）。IaaS 要求底层基础架构具有高可用性，以确保满足单个(或组)VM 的可用性服务级别协议(SLA)。单个虚拟机不包含任何可用于容忍基础架构故障的冗余。相比之下，SaaS 通常是提供商为容错而设计的分布式系统，最大限度地减少了对高可用性基础设施的依赖。因此，将 SaaS 称为软件冗余负载，将所有其他工作负载(包括 IaaS VM)称为非冗余负载。

提供商部署各种不同硬件配置(例如，计算群集、GPU 群集、存储群集)以满足客户需求。根据 Flex 中的要求，不同的硬件配

置对降低功耗有不同的支持。例如，服务器硬件通常包括 CPU/内存功率上限机制，如英特尔的运行时平均功率限制(Runtime Average Power Limit，RAPL)。RAPL 通过限制 CPU 频率(可能还有电压)以及(如果需要)内存访问来降低功耗。然而，其他耗电硬件(例如，GPU、磁盘阵列)可能没有类似的功能。因此，将硬件分为含功率限制功能硬件和不含功率限制功能硬件两类。

3.3 超额预购和功率封顶

数据中心的传统做法是限制持续功率负载为 CB 和变压器最大额定值的 80%，以避免损坏电力基础设施。例如，一个 30A 三相断路器每相只能加载到 24 A。在电力结构冗余度为 2N 的情况下，当一个电源故障，其电力负载转移到其他电源上，则剩余电源需要承担两倍的正常运行时负载。服务器电源连接必须确保这个加倍的负载同样不超过 CB 和变压器最大额定值的 80%。例如，在一个 2N 冗余 30A 电力供应中，每个负载的每个电源需要限制为 12A（30A 的 40%）以确保故障期间的总负载限制为 24A (80%)。以上传统电力预算将导致极大的电力浪费，所以现在常用的做法是结合超额预购和功率封底来提高电力利用率。

超额预购指的是让服务器在运行期间对每一个供电侧使用超过上述计算得出的限制功率。例如，在上述例子中，运行时每一个电源都限制为 24A（额定值的 80%），而不考虑故障期间的单侧电力设备总功率（额定值的 160%）超标。

功率封顶则解决的是数据中心超额预购导致的故障期间电源功率超标问题。当一个电源发生故障，冗余电源上的 CB 过载到 160%，此时需要通过功率限制将功率降低在安全性以下。通常的做法是节流 CPU（调整其电压和频率）和其他组件以强制执行功率限制，保证在 CB 跳闸时间内和 UPS 容差时间内将功率降低至安全值（80%）。

3.3 A Scalable Priority-Aware Approach to Managing Data Center Server Power

3.3.1 主要贡献

文章 1 发现负载在冗余电源上的分布不平衡，以及负载具有不同优先级。基于此，文章 1 提出了基于全局负载优先级的 CapMaestro 电源管理器。文章 1 主要有以下关键贡献点：

（1）具有冗余电源的服务器通常不会从每个电源相同的负载，并且这种不平衡的负载在功率封顶期间会对功率预算产生负面影响。

（2）CapMaestro 提出第一个用于数据中心电源管理的全局优先级感知算法，使用分布式、协调的电源控制器以强制执行电源电力基础设施中所有级别电力设备的限制。

（3）CapMaestro 提出第一个适用于具有冗余电源的服务器闭环反馈功率控制器。

3.3.2 电源控制树

CapMaestro 使用电源控制树来反映电力基础设施的层次结构。图 6 显示了树中的控制器映射到物理基础设施组件。在树的底部，上限控制器（capping controllers）使用内置服务器功率上限机制限制功率。在树的非底层，使用电力转移控制器（shifting controller）在电力输送节点之间制定该级别的功率预算。实际部署中，每个物理节点复制一份电源控制树，每个服务器有单独的跨多树共享的上限控制器。

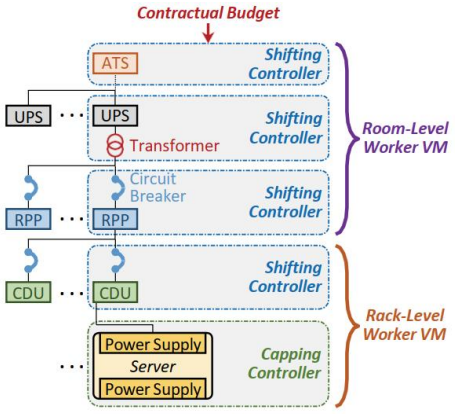


图 6 映射物理设备的电源控制树

3.3.3 全局优先级的功率封顶策略

CapMaestro 的全球优先级功率上限算法在控制树上分配功率预算，尊重数据中心合同预算和多个级别的 CB 和变压器电力限制，同时安全地尝试满足尽可能多的每台服务器的电力需求需求。对于非冗余和 N+N 个冗余配电基础设施，每个控制树独立运行该算法。该算法迭代运行，每次迭代由两个阶段组成：指标收集阶段和预算编制阶段。

在指标收集阶段，每个电力转移控制器来自其子节点的接收功率分配请求和其他指标。这些指标按优先级值分组，对应于子节点下的服务器上运行的工作负载。电力转移控制器按优先级值聚合它的子级的所有指标，并发送给上游父节点。

在预算编制阶段，每个电力转移控制器从其父节点接收其功率预算，然后根据分配的功率预算及其下游子节点的优先级计算并发送其子节点的功率预算。在底部，每个封顶控制器接收每个电源的单独预算，并对应的服务器设置功率上限。

3.3.4 指标收集

CapMaestro 在第 i 层（封顶控制器为第 0 层）每个节点上计算以下指标：

$P_{cap\_min}(i, j)$：第 i 层中优先级为 j 的负载节点的最小功率预算。

$P_{demand}(i, j)$：第 i 层中优先级为 j 的负载节点的总功率需求预算。

$P_{request}(i, j)$：第 i 层中优先级为 j 的负载节点请求的电源预算。

$P_{constraint}(i)$：第 i 层中优先级为 j 的负载节点的最大安全功率。

指标的计算在封顶控制器和电力转移控制器之间有所不同。在封顶控制器中，指标计算如下：

$$\begin{aligned} P_{cap\_min}(1, \mathrm{j}) &= r \times P_{cap\_min}(0) \\ P_{demand}(1, \mathrm{j}) &= r \times \max\{P_{demand}(0), \\ &\quad P_{cap\_min}(0)\} \\ P_{request}(\mathrm{j}) &= P_{demand}(\mathrm{j}) \\ P_{constraint} &= r \times P_{cap\_max}(0) \end{aligned} \quad (1)$$

其中，其中 r 是该电源承担的服务器负载的比例供应，比如一个服务器有 M 个电源，则$\mathrm{r} = \frac{1}{M}$。$P_{cap_max}(0)$和$P_{cap_min}(0)$是服务器的可控交流电源预算最大值和最小值。$P_{demand}(0)$是运行工作负载的电量服务器以全性能消耗。

在电力转移控制器中，指标计算方式如下：

$$\begin{aligned} &①\, P_{cap_{\min(i,j)}} = \sum_k P_{cap\_min_k}(i-1, j) \\ &②\, P_{demand}(i, j) = \sum_k P_{demand}(i-1, j) \\ &③\, P_{request}(i, j) = \min\{P_{limit} - \\ &\sum_{h>j} P_{request}(i, h) - \sum_{l<j} P_{cap_{\min(i,l)}}, \\ &\sum_k P_{request_k}(i-1, j)\} \\ &④\, P_{constraint}(i) = \\ &min\{P_{limit}, \sum_k P_{constraint_k}(i-1,)\} \end{aligned} \quad (2)$$

其中 k 是节点的子节点数量，j，h 和 l 都代表负载优先级，P_{limit} 是电力转移控制器的功率限制。

3.3.5 预算编制

每个电力转移控制器的预算编制阶段中，分配其子节点之间的功率预算分为以下四步：

（1）为每个子节点分配 P_{cap_min}的电力。

（2）从高优先级进行迭代，进一步分配每个子节点请求的额外功率，即 $P_{request} - P_{cap\_min}$。如果控制器预算中剩余的电量不够，转步骤 3；否则，完成所有操作后转到步骤 4。

（3）对于电力预算无法满足的最后一个优先级 j，按比例给出每个子节点分配剩余预算，即 $P_{demandt} - P_{cap\_min}$。

（4）如果完成后还有剩余预算，在保证子节点不超过 $P_{constraint}$的情况下继续分配预算。

同时在预算编制中穿插搁浅电力优化（stranded power optimization，SPO），可以有效减少搁浅电力。SPO 原理为先判断哪部分预算可能产生搁浅电力，并由此削减对应的电力预算，然后重复上述预算分配步骤。

3.3.4 实验

在 CapMaestro 的验证中，使用虚拟机

实现数据中心管理器，包括：机架级 VM Worker 和间级 VM Worker，CapMaestro 的控制周期为 8S。为了验证全局优先级的作用，设置了两个对照组：基于无优先级和本地优先级的电力预算，电源架构如图 1 所示。

表 2 基于不同优先级下的电力预算

Server	SA	SB	SC	SD
Priority (H = high, L = low)	H	L	L	L
Power Demand (W)	420	413	417	423
Budget with No Priority (W)	314	306	311	316
Budget with Local Priority (W)	344	274	314	317
Budget with Global Priority (W)	419	276	275	275

表 2 显示了 CapMaestro 的能力，高优先级服务器可以从全局中抽调电力。基于本地优先级的电源预算调度则只能从 SB 向 SA 抽调。

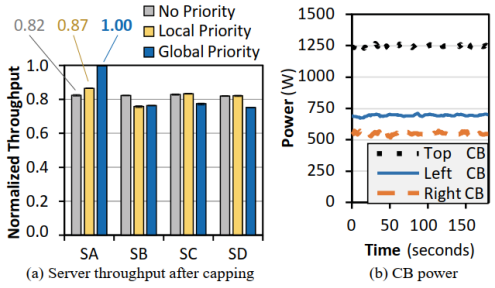


图 7（a）不同服务器吞吐率（b）CBs 运行时功率

图 7（a）显示全局优先级方法有效地增加高优先级服务器的吞吐量，且对低优先级影响不大。图 7（b）显示了各个 CBs 的运行时功率，表示该系统是在安全功率下运行的。

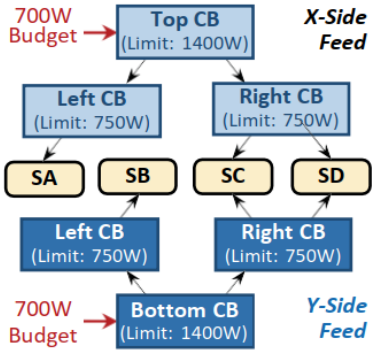


图 7 服务器电源架构

为了演示 CapMaestro 的 SPO，实验将四台服务器（SA、SB、SC 和 SD）连接到两个冗余电源（X 侧和 Y 侧），如图 7a。服务器 SA 具有高优先级，而其他三个服务器的优先级较低。 断开 Y 侧的供应 SA 和 SB 的 X 侧供应。表 3 显示了搁浅电力优化过程， SC 和 SD 的搁浅电力被分配到 SB 上，其中/的两侧表示服务器从两个供电侧获得的电力。

表 3 不同服务器的电源预算以及实际消耗电量

Server		SA	SB	SC	SD
Priority (H = high, L = low)		H	L	L	L
Demand (W)		414	415	433	439
Global Priority	Budget	415/0	0/346	152/*164*	132/*187*
w/o SPO (W)	Consumption	413/0	0/348	156/*137*	135/*158*
Global Priority	Budget	416/0	0/413	152/132	132/155
w/ SPO (W)	Consumption	413/0	0/412	153/134	133/156

实验使用典型 Google 数据中心负载数据和满功率极端情况来测试服务器部署数量。从图 8 的结果来看，CapMaestro 在极端情况下的服务器部署数量明显提高，且能保持 92.3%的较高服务器支持率。

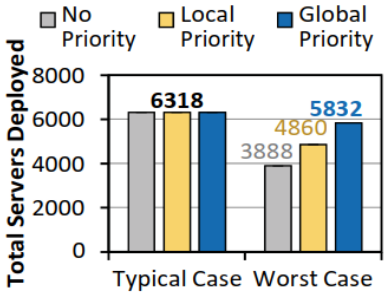


图 8 服务器部署数量对比

3.4 Prediction-based power oversubscription in cloud platforms

3.4.1 主要贡献

文章 2 有以下几个关键贡献点：

（1）一个预测 VM 重要性的算法和机器学习模型，以及一个预测 VM 对服务器利用率的模型。

（2）一个最小化功率封顶事件数量和影响的基于预测的 VM 放置策略。

（3）实现故障期间充分保护重要 VM 的性能。

（4）对超额预购的比例进行提升。

3.4.2 重要 VM 选择算法

在预测 VM 重要性上，首先需要说明的是，VM 的重要性由是否为用户交互 VM 来确定，因为这部分 VM 往往服务于付费用户。之前的工作通过分析用户交互性操作的时间规律来判断 VM 的重要性，即是否判断负

载的 CPU 利用率具有 24 小时周期性。具体地，使用快速傅里叶算法（Fast Fourier Transform, FFT）分析 CPU 利用率时序图来判断负载重要性。文章 2 分析了 FFT 方法不准确的三点原因：

（1）面向用户的负载被一段时间的恒定或随机负载打断，则可能被误判为非用户负载。

（2）用户负载的昼夜规律往往有递增或递减变化趋势。

（3）一些具有小周期（1 小时，4 小时等）特点的负载也会被判断为具有 24 小时周期特点。

基于以上问题，文章 2 设计了一个新的识别算法，具体步骤如下：

（1）对 CPU 利用率时序图进行去趋势化和标准化。

（2）对 CPU 利用率每 30 分钟取平均值作为采样数据。

（3）创建一个时序模板，然后计算一天内模板与实际利用率的误差。

（4）重复步骤 2 和 3，计算 8 小时偏差和 12 小时偏差，然后用 24 小时偏差除 8 小时偏差（称为 Compare8）和 24 小时偏差除 12 小时偏差（称为 Compare12），如果值趋近于 0，则代表该负载属于用户负载。以上算法计算得出的值可以作为机器学习的训练集标签。

3.4.3 重要 VM 预测算法

以上算法可以通过历史 VM 运行信息判断 VM 重要性，形成带标签的 ML 训练数据，用来训练 ML 模型以对到达的 VM 进行面向用户或非面向用户的类型判断。具体来说，我们训练一个带标签的随机森林 ML 模型，输入的数据包括：面向用户的虚拟机的订阅比例、至少运行 7 天的 VM 的比例、虚拟机总数订阅、每个 CPU 利用率下的 VM 的占比、VM 的平均 CPU 利用率和第 95%的 CPU 利用率（虚拟机生命周期内的实际从低到高排序的第 95%的 CPU 利用率）、VM 的需求内核数量和内存大小以及到达的 VM 的类型。

3.4.4 CPU 预测算法

由于准确预测利用率很难，因此将 CPU 利用率分成四个区间（Bucket）： 0%-25%、26%-50%，以此类推。和上面一样，根据之前 VM 执行产生的标签预测 和 VM 生命周期内 95%的 CPU 利用率作为输入，训练一个随机森林模型，用来预测到来的 VM 的 CPU 利用率的所在区间。

3.4.5 VM 放置算法

一个好的放置算法需要实现以下目标：

（1）负载分布平衡，减少功率封顶事件的发生。

（2）不影响重要 VM 性能的情况下降低功率。

（3）减少 VM 部署失败的次数。

根据以上要求，文章 2 提出了一个 VM 放置算法如下：

Algorithm 1 Criticality- & utilization-aware VM placement

1: **function** SORTCANDIDATES(V, ζ)
　　▷ V: VM to be placed, ζ: list of candidate servers
2: 　$\omega \leftarrow V^{PredictedWorkloadType}$
3: 　**for** c_i in ζ **do**
4: 　　$\kappa_i \leftarrow$ SCORECHASSIS($c_i.Chassis$)
5: 　　$\eta_i \leftarrow$ SCORESERVER(ω, c_i)
6: 　　$c_i.score \leftarrow \alpha \times \kappa_i + (1-\alpha) \times \eta_i$
7: 　**return** ζ.SORTDESC($c_i.score$)
8: **function** SCORECHASSIS(C)
9: 　**for** n_i in $C.Servers$ **do**
10: 　　**for** v_j in n_i^{VMs} **do**
11: 　　$\rho^{Peak} \leftarrow \rho^{Peak} + v_j^{PredictedP95Util} \times v_j^{cores}$
12: 　　$\rho^{Max} \leftarrow \rho^{Max} + n_i^{cores}$
13: 　**return** $1 - \left[\frac{\rho^{Peak}}{\rho^{Max}}\right]$
14: **function** SCORESERVER(ω, N)
15: 　**for** v_i in N^{UF_VMs} **do**
16: 　　$\gamma^{UF} \leftarrow \gamma^{UF} + v_i^{PredictedP95Util} \times v_i^{cores}$
17: 　**for** v_i in N^{NUF_VMs} **do**
18: 　　$\gamma^{NUF} \leftarrow \gamma^{NUF} + v_i^{PredictedP95Util} \times v_i^{cores}$
19: 　**if** $\omega = UF$ **then**
20: 　　**return** $\frac{1}{2} \times \left(1 + \frac{\gamma^{NUF} - \gamma^{UF}}{N^{cores}}\right)$
21: 　**else**
22: 　　**return** $\frac{1}{2} \times \left(1 + \frac{\gamma^{UF} - \gamma^{NUF}}{N^{cores}}\right)$

函数 *SORTCANDIDATES* 用来返回当一个 VM 到来时，最适合放置的服务器列表。其中打分函数由当前放置的机架分数以及服务器分数加权得到（第 6 行）。

SCORECHASSIS 函数通过计算当前机架的峰值功率ρ^{Peak}以及最大功率ρ^{Max}的比值来判断机架空闲程度（第 13 行）。函数返回结果越高，表示机架约空闲，得分越高。

SCORESERVER 函数作用是协调当前服务器上用户 VM（重要 VM）和非用户 VM 的数量。当部署一个面向用户的 VM 时，计算服务器上非面向用户的 VM 比面向用户的 VM 的利用率差值（第 19-20 行），若大于 0

则表示当前非面向用户的 VM 负载多，部署面向用户的 VM 更有利。

3.4.6 实验

图 9 显示了 Compare8 和 Compare12 对负载的区分能力可视化。如图所示，Compare8 可以将用户负载和非用户负载（红色和蓝色代表的负载）很好的区分开（两种负载在 Compare8 上的投影在 0.72 左右区分开）。而 Compare12 的效果则不好，因为两种负载在 Compare12 上的投影存在大量重合部分。所以只使用 Compare8 作为重要 VM 选择算法。

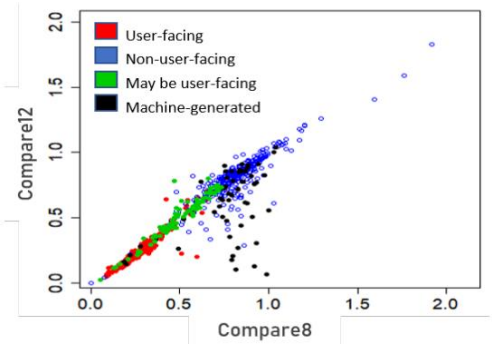


图 9 Compare8 和 Compare12 算法比较

由于目标是想要保护面向用户的 VM，我们必须实现此类的高召回率，因为召回率表示概率正确识别这些 VM。表 4 显示了不同算法的精确率和召回率，对于两个高召回率目标（0.99 和 0.98）的向用户工作负载，文章 2 的模式识别算法实现了目标比同类产品高得多召回的精度。

表 4 重要 VM 选择算法对比

Technique	Recall target	Recall achieved	Precision achieved
Pattern-matching	99%	99%	76%
ACF	99%	99%	54%
FFT	99%	99%	48%
Pattern-matching	98%	98%	77%
ACF	98%	98%	56%
FFT	98%	98%	50%

表 5 显示了随机森林 (RF) 和梯度提升 (GB) 模型召回(R)、精度 (P) 和高置信度预测的准确性。该表显示我们的关键性模型达到了 99%面向用户的 VM 召回率，这对于保护这些虚拟机非常重要。GB 模型实现相似的准确性，但具有较少的高置信度，且在中间两个区间（Bucet2-3）的预测上召回率较低。

表 5 随机森林 (RF) 和梯度提升 (GB) 模型召回率和准确度

Prediction	Model	% High Conf.	Bucket 1 R — P	Bucket 2 R — P	Bucket 3 R — P	Bucket 4 R — P	Accuracy
Criticality	GB	99%	67% — 77%	99% — 99%	NA	NA	98%
	RF	99%	69% — 78%	99% — 99%	NA	NA	98%
P95 util	GB	68%	95% — 85%	47% — 77%	51% — 79%	94% — 80%	82%
	RF	73%	93% — 87%	61% — 76%	65% — 81%	92% — 83%	84%

从图 10 中可以看出，在系统功率上限为 250W 的情况下，进行 VM 预测并限制非重要 VM 的性能可以达到限制所有服务器性能相同的效果，均保持了数据中心运行时安全。

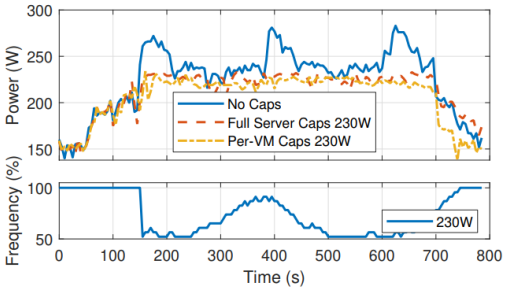


图 10 服务器电力时序图

从图 11 可以看出，在功率限制不断降低下，面向用户的 VM 性能表现受到的影响

在本系统方法中远小于全服务器统一限制方法，而非用户 VM 的性能降低更多，从而保护了重要 VM 的性能。

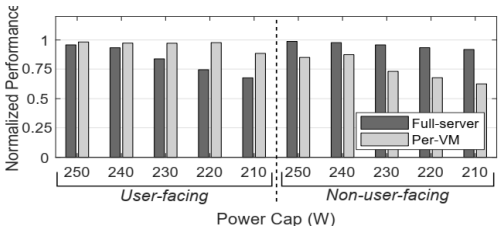


图 11 不同功率限制下的 VM 性能

如表 6 所示，最先进的不具有 VM 预测功能的方法实现了 6.2% 的超额订阅。相比之下，文章 2 的方法（表中第 3、4 行）几乎可以翻倍超额认购，达到了 12% 以上超额订阅，且对面向用户的 VM 性能影响小。假设一个 128MW 的数据中心和 10 美元/W 的基础设施电力成本，12.1% 的超额订阅提升相当于节省 1.549 亿美元。

表 6 不同电力供应策略的结果对比

Approach	Chassis budget delta (%)	Savings ($10/W)
Traditional	0	0
State of the art	6.2%	$79.4M
Predictions for all VMs, no UF impact	11.0%	$140.8M
Predictions for all VMs, minimal UF impact	12.1%	$154.9M

3.5 Flex: High-Availability Datacenters With Zero Reserved Power

3.5.1 主要贡献

文章 3 的主要贡献有以下几点：

（1）提出了高可用、零预留功率数据中心的概念，并证明它可以通过基础设施支持和分布式软件的结合实现。

（2）设计并实现了 Flex，这是一个用于管理零预留电力数据中心的系统。Flex 包含一个离线组件，负责用于巧妙地放置工作负载和在线组件动态管理电源可用性和性能。

（3）通过真实工作负载的模拟和实验对 Flex 进行评估，以证明 Flex 有效地减少了预留电力和搁浅功率。

3.5.2 Flex-Offline 系统

Flex-Offline 目标是将所有请求的负载合理部署在服务器中，将搁浅的电力降至最低，同时确保在任何维护情况下在（最坏的情况即电力 100%利用率)可以削减足够的电力保证系统安全。其中，削减空间来自软件冗余机架中全部分配的功率，和非冗余但可设置上限的机架中已分配的功率和最低运行功率之间的差值。文章 3 将 Flex-Offline 建模为一个基于整数线性规划(ILP)的自动化工作负载分配算法。先定义几种符号：$\hat{d}$ 表示负载请求集合，$\hat{u}$代表UPS，$\hat{p}$代表PDU，$Map(p \in \hat{p}) \rightarrow (u_1, u_2) \mid u_1, u_2 \in \hat{u}$ 表示 PDU 与 UPS 的连接情况，$P_{d,p}$表示一个负载 d是否部署在p下，其中$d \epsilon \hat{d}$，$p \epsilon \hat{p}$。ILP 由以下五条规则组成：

$$\forall d \in \hat{d}, \sum_{p \in \hat{p}} P_{d,p} \leq 1 \quad (3)$$

公式 3 表示一个负载至多部署一次。

$$\forall u \in \hat{u}, \left[Load_u = \sum_{\substack{d \in \hat{d} \\ p \in \hat{p}}} 0.5 \times P_{d,p} \times Pow_d | u \in Map(p) \right] \leq Capacity_u \quad (4)$$

公式 4 表示一个电源 u 上的所有负载加起来不能超过额定功率。

$$CapPow_d = \begin{cases} 0 & d \in \text{Software-redundant} \\ FlexPow_d & d \in \text{Non-redundant, cap-able} \\ Pow_d & d \in \text{Non-redundant, non-cap-able} \end{cases} \quad (5)$$

公式 5 表示一个负载 d 在故障情况下的最大功率降低空间。

$$\begin{aligned} &\forall f \in \hat{u}, u \in \hat{u}, f \neq u, \\ &\left(\sum_{\substack{d \in \hat{d} \\ p \in \hat{p}}} 0.5 \times P_{d,p} \times CapPow_d | u \in Map(p) \right) + \\ &\left(\sum_{\substack{d \in \hat{d} \\ p \in \hat{p}}} 0.5 \times P_{d,p} \times CapPow_d | u \in Map(p), f \in Map(p) \right) \leq Capacity_u \end{aligned} \quad (6)$$

公式 6 表示故障期间，负载转移的目标服务器的原有负载加上转移来的负载不能超过额定功率。

$$StrandedPow = \sum_{u \in \hat{u}} Capacity_u - Load_u \quad (7)$$

公式 7 表示系统整体目标是最小化搁浅电力。

3.5.3 Flex-Online 系统

由于超额预算的存在，电力系统可能出现超过 100%额定功率的情况。这种情况下需要在 UPS 容差范围内（通常是 10s）降低功率，所以可靠、高保真和低延迟的电力遥测在故障时的重要性不言而喻。Flex 通过设计了高度冗余的遥测系统实现该目的。

Flex 的运行决策系统的目的是当有 UPS 运行功率超过极限功率时，通过该系统选择关闭/节流策略，同时最小化对当前负载的影响。Flex 的运行决策算法分为以下三部分：

（1）估计可从候选机架恢复的电量，作为降低空间。

（2）循环遍历每一个机架，计算关闭/节流动作对负载的性能影响值。

（3）持续监控系统状况，知道功率恢复正常。在恢复正常后，重新开放被关闭的机器或解除机器节流。

其中的性能影响通过微软总结的影响函数曲线计算，如图 12 所示。其中，（A）代表一个典型的非冗余但有节流能力的工作负载；(B) 是一个软件冗余工作负载，在关闭一定机器前对性能没有影响；C 是一个软件冗余的工作负载，在关闭一定机器后性能逐步下降。

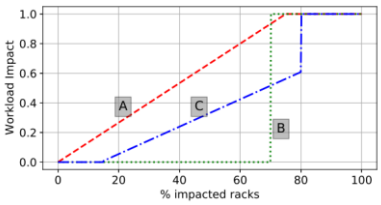


图 12 不同负载下的影响函数

3.5.4 实验

实验按照 4N/3 冗余度的电源架构进行搭建，采用微软典型数据中心部署情况。实验评估了两个数值：搁浅电力占比以及节流平衡性。实验的对比策略有：随机放置策略、平衡轮询放置策略和 Flex-Offline 策略。其中在 Flex-Offline 策略的测试中，根据负载所需电力大小分了三个类型：Flex-Offline-Short，使用 33%电力的负载；Flex-Offline-Long，使用 66%电力的负载和 Oracle，全负载。

如图 13 所示，是不同策略下的搁浅电力占比，可以得出：在微软的新电力结构中，所有策略的搁浅功率占比都小于 10%，且 Flex 的方案均显示出更低的搁浅功率。

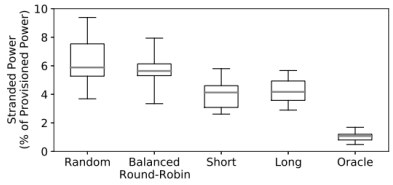


图 13 不同策略下的搁浅电力占比

如图 14 所示是不同策略下的节流平衡性实验结果。其中平衡性度量为一次节流事件中中节流的最大值减去节流最小值。图 14 显示了 Flex 有更好的节流平衡性，即可以从集群中更平衡地抽取电力资源。

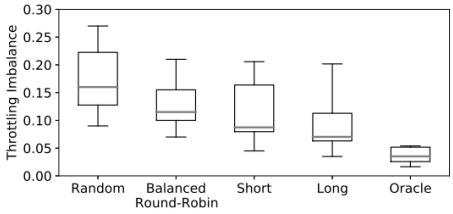
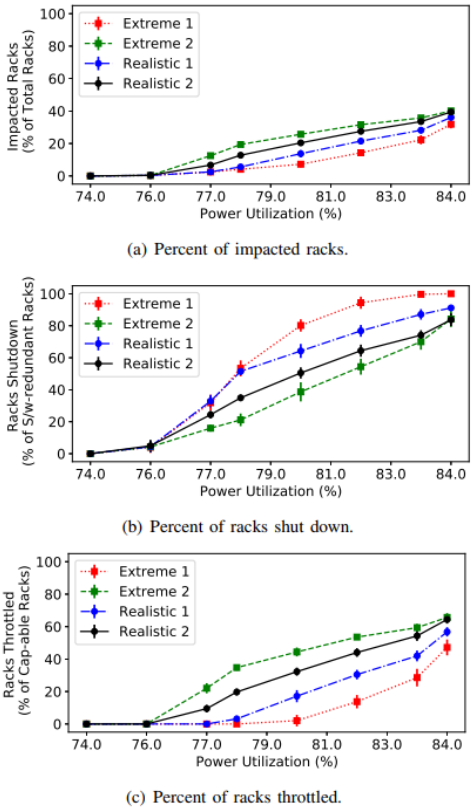


图 14 不同策略下的节流平衡性



(a) Percent of impacted racks.

(b) Percent of racks shut down.

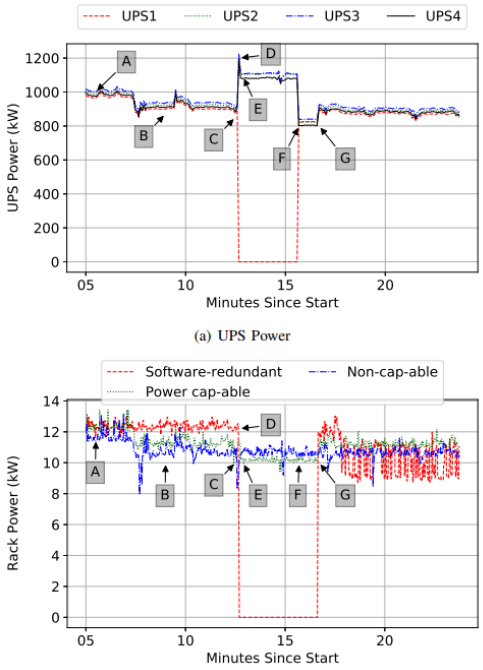
(c) Percent of racks throttled.

图 15 Flex 运行决策系统的运行过程

如图 15 所示是 Flex 运行决策系统的运

行过程中受影响机架、关闭的机架和节流的机架的数量。可以看出，当冗余机架关闭越多的情况下，需要节流的机器越少，反之亦然。而真实情况（黑色和蓝色曲线）均处于两中极端情况（绿色和红色曲线）之间，意味着 Flex-Online 成功地根据优先级确定其纠正措施工作负载的性能和可用性需求维护事件，Flex 可以根据不同负载的需求做出对应的维护决策。

如图 16 所示是 Flex 端到端测试运行决策时的系统功率曲线。在该曲线中可以看出 Flex 在出现故障（C 点），系统功率超过额定功率（D 点）后，快速且有效的功率调节功能。



(a) UPS Power

图 16 Flex 运行决策时的系统功率曲线

4 总结与展望

文章 1 提出的 CapMaestro 是一种新的分布式电源管理机制，它有效地使用服务器电源上限来管理整个数据中心的电源。其最主要特点是实现了负载全局优先级感知，据此改进超额预算，部署更多服务器。文章 2 从 VM 的角度对数据中心电源管理机制进行改进。其最主要特点是设计了一个 VM 重要性选择算法，以此建立了预测 VM 重要性的 ML 训练集，得到一个可以判断 VM 重要性的 ML 模型和一个预测 VM 资源利用率的模型。文章 2 依据这两个模型提高了两倍的超额预算比例，且可以保护重要 VM 的性能。文章 3 引入了零预留电力数据中心并且设计实现 Flex 系统来管理它们。Flex 通过将负载放置建模为整数线性规划问题最小化搁浅电力，通过改进电力监控系统和运行时决策算法来保证系统在超额预算情况下的运行安全。

三篇文章都在经过实际部署后中提出大致相同的展望：

（1）升级配套的基础设施。更高超额预算下上游供电设施的安全稳定需要解决。

（2）考虑多方面优化因素。现有优化都只基于电力成本，后续还需要考虑占地、冷却等资源的成本。

（3）设计新的收费模式。现有系统都全力保障面向用户的负载性能，但往往有的用户没有高性能需求，可以通过降低用户费用激励其工作负载更低的性能和可用性要求。

参考文献

[1] Y. Li, C. R. Lefurgy, K. Rajamani, M. S. Allen-Ware, G. J. Silva, D. D. Heimsoth, S. Ghose, and O. Mutlu, “A Scalable Priority-Aware Approach to Managing Data Center Server Power,” in Proceedings of the International Symposium on High Performance Computer Architecture, 2019

[2] A. Kumbhare, R. Azimi, I. Manousakis, A. Bonde, F. Frujeri, N. Mahalingam, P. Misra, S. A. Javadi, B. Schroeder, M. Fontoura, and R. Bianchini, “Prediction-based power oversubscription in cloud platforms,” in Proceedings of the USENIX Annual Technical Conference, 2021

[3] Zhang, Chaojie, Alok Gautam Kumbhare, Ioannis Manousakis, Deli Zhang, Pulkit A. Misra, Rod Assis, Kyle Woolcock, et al. “Flex: High-Availability Datacenters With Zero Reserved Power.” In 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), 319–32. Valencia, Spain: IEEE, 2021