doi:10.3969/j.issn.1000-436x.2014.02.025

云计算中的服务可用性保障机制

沈时军[1]，刘欣然[1,2]，张鸿[1]，朱春鸽[1,2]
（1. 国家计算机网络应急技术处理协调中心，北京 100029；2. 北京邮电大学 信息安全中心，北京 100876）

摘 要：提出了一种基于滑动窗口的资源预留 SWRR(sliding window based resource reservation)算法，它将预留资源在整个资源池中所占的比例称为窗口。窗口的“滑动”包含 2 层含义：1）窗口大小动态变化；2）窗口中的资源动态刷新。SWRR 已被应用于一个大型的云计算应用平台。实验数据表明，SWRR 通过合理资源预留，在兼顾所有任务调度的基础上，可为特定用户提供有效的服务可用性保障。
关键词：云计算；资源预留；滑动窗口；服务可用性
中图分类号：TP393 **文献标识码**：A **文章编号**：1000-436X(2014)02-0202-05

Guaranteeing service availability in cloud computing

SHEN Shi-jun[1], LIU Xin-ran[1,2], ZHANG Hong[1], ZHU Chun-ge[1,2]
(1. National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China;
2.Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: A sliding window based resource reservation (SWRR) algorithm was proposed, in which a window is defined as the proportion of the reserved resources among all. The sliding of the window is twofold, adaptively window resizing, and dynamically window content refreshing. SWRR was applied to a large cloud computing platform. Experiments reveal that SWRR can provide high service availability for specific users while taking all users into account by proper resource reservation.
Key words: cloud computing; resource reservation; sliding window; service availability

1 引言

云计算通过虚拟化技术动态整合 IT 基础设施与平台资源，利用互联网络为企业与个人用户提供按需的、廉价的计算、存储、软件、平台等服务。近年来，随着国外 Amazon EC2、Google App Engine、Microsoft Azure 及国内阿里云等一系列云计算产品的成功以及 OpenStack、CloudStack、Eucalyptus、Open Nebula 等许多开源云平台的推广，云计算已被认为是继大型计算机、个人计算机、互联网之后的又一次 IT 产业革命。

安全问题一直是云计算面临的最重要挑战之一，如数据泄密、服务可用性、拒绝服务攻击等[1,2]。虽然云计算因其良好的可扩展性被称为弹性计算，但当大量服务请求在同一时间到达时仍可能耗尽云计算平台中的所有资源。此时若有新的请求，特别是重要用户的请求到达时，将无法得到满足。考虑到投资回报，绝大多数云计算平台都不会长期维持大量的空闲资源，因此这种服务可用性的威胁无论在商业云还是私有云中都是常见的[3,4]。

为了应对这种服务可用性威胁，价格调控与资源预留是 2 种主要的手段。价格调控存在于绝大多数商业云中。服务商根据当前的资源紧张程度动态调整服务价格，使供需保持相对的平衡，减少资源耗尽的危险。例如在 Amazon EC2 中，服务商推出了按需支付(on-demand instance)服务、预约支付(reserved instance)服务、竞价支付(spot instance) 服务 3 种不同的收费服务模式[5]。资源预留策略存在

收稿日期：2013-05-02；**修回日期**：2013-11-04
基金项目：国家重点基础研究发展计划(“973”计划)基金资助项目(2011CB302605)
Foundation Item: The National Basic Research Program of China (973 Program) (2011CB302605)

于各种云平台中。在一些私有云中，由于服务是免费的，价格调控无法进行，资源预留成为保障重要用户服务可用性的主要手段[6]。由于用户需求的不可知性，预留的资源过多可能造成浪费，预留的资源过少又可能无法满足用户需求。在文献[7]中，用户提前向云计算服务商预订可能用到的资源，服务商收集这些用户的需求后，通过资源新增、任务调度等手段，保证特定时间的资源可用性。由于用户通常很难回答“什么时间、需要多少资源”，这种策略可能有一定的局限性。

文献[6]指出，“在没有价格调控的情况下，如何既保证重要用户的请求得到满足，又不因资源过度预留而造成浪费，是云计算中的重要挑战之一”。本文提出了一种云计算中的服务可用性保障机制——基于滑动窗口的资源预留(SWRR, sliding window based resource reservation)算法。它将云平台用户提交的任务分为普通任务与重要任务(VIP 任务)。SWRR 的基本思想是：在云计算平台中为 VIP 任务预留一定数量的优秀资源，并将这部分优秀资源在整个资源池中所占的比例称为窗口。窗口的“滑动”包含 2 层含义：1）窗口大小动态变化，即在 VIP 任务增加时扩大窗口以保证优先调度，在 VIP 任务减少时缩小窗口以减少资源浪费；2）窗口内容动态刷新，即每隔一段时间会评估窗口中的预留资源，淘汰差的资源，引入新的优秀资源。

SWRR 已被应用于一个大型的云计算应用平台 iVCE 中，如图 1 所示。目前，该平台包含分布于全国各地的 300 多台物理机，可支持每天百万级的任务吞吐量。平台中用户每天提交的任务数随时间的波动很大且很难预测，在高峰时段大量的用户请求可能相互抢占资源，平台的服务可用性变差。为了体现差分服务的特点，iVCE 授予某些用户提交 VIP 任务的权限，并采用 SWRR 优先调度。实际运行结果表明，SWRR 可以保证 VIP 任务的服务等待时间（一般小于 5 s）远小于普通任务（平均约 60 s），保障重要用户的服务可用性。

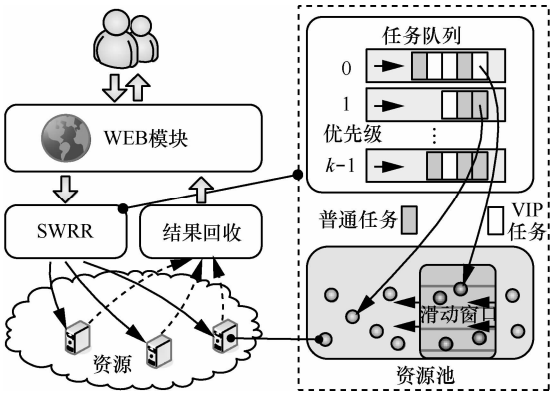


图 1　SWRR 算法示意

由于在实际环境中直接调试 SWRR 参数可能影响平台稳定性，本文实现了一个 SWRR 仿真器，并通过多方面的分析比较，论证 SWRR 在资源预留、保障重要用户服务可用性方面的优势。

2　基于滑动窗口的资源预留算法

2.1　资源的定义

云计算平台中可用的资源集合记为 $\Phi=\{r_i \mid i = 1, 2,\cdots, n\}$，该集合动态变化。对每一个资源 r_i，定义资源评价函数 $V(r_i)$为 r_i在过去一段时间内的任务执行成功率，$V(r_i)$越大表示 r_i越优秀。

SWRR 算法将 Φ 划分为普通资源池 Φ_{base}与预留资源池 Φ_{resv}，分别存放普通资源与预留资源。初始时，$\Phi_{\text{base}} = \Phi$，$\Phi_{\text{resv}}$为空。

2.2　任务的定义

云计算平台中需调度的任务集合记为 $\Psi=\{t_j \mid j = 1, 2,\cdots, m\}$，该集合动态变化。对每一个任务 t_j，令 $I(t_j)$表示任务 t_j的重要程度，若 $I(t_j) = \text{true}$，表示 t_j为 VIP 任务；否则，为普通任务。

任务分为 z 个优先级，记为 $Z = \{1, 2,\cdots, z\}$。对每一个任务 t_j，令 $P(t_j)$表示任务 t_j的优先级，则优先级为 k 的任务集合记为 $\Psi_k = \{t_j|P(t_j) = k, k\in Z\}$。任务集合 Ψ_k中的任务按先入先出（FIFO）算法组成队列，任务从队首开始调度。

2.3　SWRR 算法

根据任务优先级的不同，用户提交的任务被保存在相应的任务队列 Ψ_k中，并按照 FIFO 的顺序进行调度。SWRR 算法包括任务调度与窗口滑动 2 个阶段，如算法 1 所示。

在算法的任务调度阶段，采用“轮盘赌”选择需要调度的任务队列 Ψ_k，并且优先级 k 越大的队列被选中的概率越大。如果任务 t_j为 VIP 任务，优先在预留资源池 Φ_{resv}中查找资源，在 Φ_{resv}查找失败后继续在 Φ_{base}中查找；如果任务 t_j为普通任务，则仅在 Φ_{base}中查找资源。标志位 f_{base}、f_{resv}反映当前的资源池状态。当 Φ_{base}资源不足时，f_{base} = false；当 Φ_{resv}资源不足时，f_{resv} = false。

在算法的窗口滑动阶段，根据标志位 f_{base}、f_{resv}调整滑动窗口大小 ω。如果预留资源池 Φ_{resv}资源

不足，则扩大滑动窗口；如果预留资源池 Φ_{resv} 资源充足并且普通资源池 Φ_{base} 资源不足，则缩小滑动窗口；否则窗口大小不变。系统预设滑动窗口的最大值 ω_{max}、最小值 ω_{min}，$\omega \in [\omega_{min}, \omega_{max}]$。在每一个调度周期结束后，根据 $V(r_i)$重新评估所有资源，并将 Φ_{resv} 中最差的部分资源与 Φ_{base} 中最优秀的部分资源互换，以保证 Φ_{resv} 始终维持最优秀的资源。

算法 1 SWRR 算法

给定资源集合 Φ，任务集合 Ψ，滑动窗口大小 ω。给定常数：滑动窗口的最大值 ω_{max}、最小值 ω_{min}，窗口调整步长 $\omega' \in (0,1)$ ，资源交换率 $\lambda \in (0,1)$。初始时，令 $\omega = \omega_{min}$。

在每一个任务调度周期，进行如下步骤。

1) 初始化标志位 f_{base} = true，f_{resv} = true。

2) 采用“轮盘赌”算法选取一个需要调度的任务队列 Ψ_k（$k \in Z$）。

3) 从 Ψ_k 中取出第一个需要调度的任务 t_j。

① 若 t_j 为 VIP 任务，即 $I(t_j)$ = true，则从 Φ_{resv} 中选取若干符合 t_j 条件的资源组成备选资源池 $\Phi(t_j)$。

若 $\Phi(t_j)$非空，从 $\Phi(t_j)$中随机选取资源 r_i，将 t_j 下发到资源 r_i 上执行，然后转步骤 4)。

若 $\Phi(t_j)$为空，设置 f_{resv} = false，并将 t_j 当成普通任务继续步骤 3)中的②。

② 从 Φ_{base} 中选取若干符合 t_j 条件的资源组成备选资源池 $\Phi(t_j)$。

若 $\Phi(t_j)$非空，从 $\Phi(t_j)$中随机选取资源 r_i，将 t_j 下发到资源 r_i 上执行，然后转步骤 4)。

若 $\Phi(t_j)$为空，设置 f_{base} = false，增加 t_j 的调度失败次数，将 t_j 放回 Ψ_k，然后转步骤 4)。

4) 循环进行步骤 2)、步骤 3)，直到本次调度时间结束，或者所有任务队列都没有需调度的任务。

5) 确定新的滑动窗口大小 ω。

① 若 f_{resv} = false，则设置 $\omega = \min(\omega_{max}, \omega+\omega')$。

② 若 f_{resv} = true 并且 f_{base} = false，则设置 $\omega = \max(\omega_{min}, \omega-\omega')$。

③ 否则 ω 不变。

6) 令 $a = |\Phi_{base}|$表示普通资源数，$b=|\Phi_{resv}|$表示预留资源数。从 Φ_{resv} 中选择 $V(r_i)$评价最差的 λb 个资源，从 Φ_{base} 中选择 $V(r_i)$评价最优的 $\max\{(a+b)\omega-(b-\lambda b), 0\}$个资源，对以上两部分资源互换。

在算法 1 中，如果找不到合适的资源，任务 t_j 调度失败后将重新被放回队列 Ψ_k 中。为了避免同一个任务不断地被选中调度，当任务 t_j 调度失败后，它将等待一定时间才能被再次调度，并且等待时间随着调度失败次数的增长而指数递增。

SWRR 算法具有如下特点。

1) 自适应地资源预留。通过滑动窗口大小调整与资源交换，始终维持合适的预留资源，保证 VIP 任务的调度。

2) 兼顾所有任务调度。通过限制滑动窗口大小 $\omega \in [\omega_{min}, \omega_{max}]$，保证普通任务的调度；通过“轮盘赌”算法，低优先级的任务也有被调度的概率。

3) 对 VIP 任务突发性激增的适应性。算法中滑动窗口调整到合适的位置需要一定的时间，为保证服务质量，若 VIP 任务暂时在 Φ_{resv} 中找不到资源时，将继续在 Φ_{base} 中寻找。

3 仿真实验

本文实现了一个 SWRR 仿真器，并通过多方面的分析比较，论证 SWRR 在资源预留、保障重要用户服务可用性方面的优势。实验中，SWRR 参数采用与云计算平台 iVCE 中相同的设置，即滑动窗口最小值 ω_{min} = 0.2、最大值 ω_{max} = 0.6，窗口调整步长 ω' = 0.02，资源交换率 λ = 0.2。在本节最后，会进一步讨论这些值的选择依据。

仿真中最大资源数$|\Phi|$=1 000，每个资源的性能在资源初始化时随机生成，该指标将影响任务的执行时间与执行成功率。实验通过一个任务生成器，每秒生成一定数目的 VIP 任务与普通任务，以模拟用户提交任务的行为。每个任务在生成时会附带一个截止时间的属性，如果任务在截止时间到达时仍没有执行完成，将标识为失败，并以此统计任务执行成功率。任务在截止之前，如果没有找到合适的资源，它将处于等待状态，并以此统计任务等待时间。

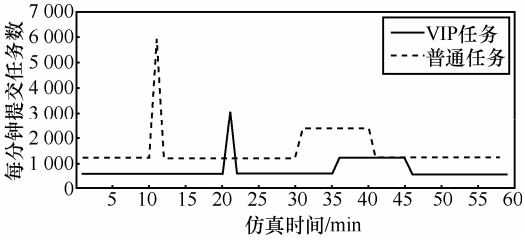
仿真器运行于 Linux 平台。实验在运行稳定后，截取其中的 60 min，并分析其结果。

3.1 SWRR 算法性能

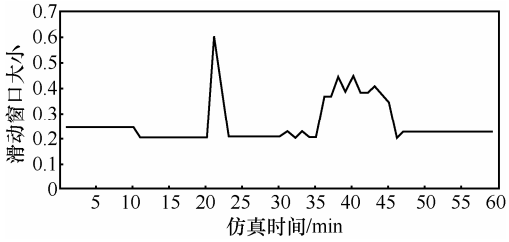
图 2（a）给出了实验中每分钟的任务提交情况。参考云计算平台 iVCE 的任务提交行为，实验分别模拟了普通任务突发性激增（第 10~11 min）、VIP 任务突发性激增（第 20~21 min）、普通任务与 VIP 任务持续高负载（第 30~45 min）的情况。

实验可以分为 5 个阶段。

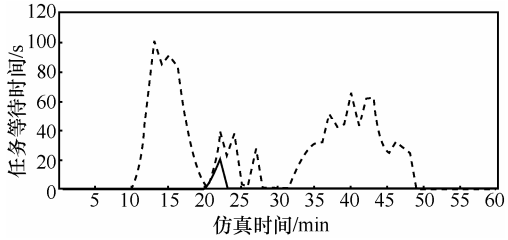
第 0~10 min 为第 1 阶段。此时系统资源足够，可处理所有的普通任务与 VIP 任务，系统处于平稳运行状态，滑动窗口 ω 约为 0.24，如图 2(b)所示。



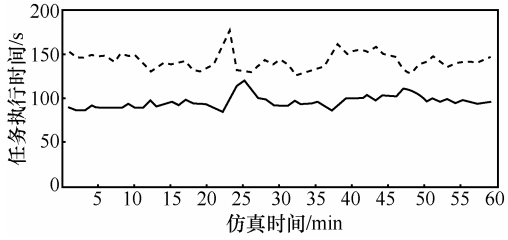
(a) 每分钟提交任务数



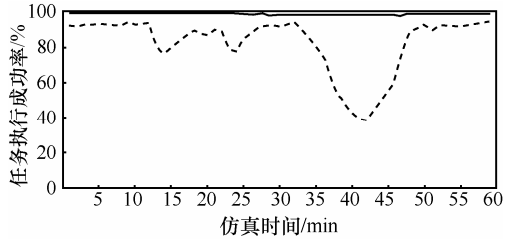
(b) 滑动窗口变化



(c) 任务等待时间



(d) 任务执行时间



(e) 任务执行成功率

图 2　VIP 任务与普通任务的运行比较

第 10~20 min 为第 2 阶段。在第 11 min，普通任务出现激增，如图 2(a)所示，普通资源池 Φ_{base} 出现资源不足，滑动窗口 ω 降为最小值 0.2，如图 2(b)所示，以尽可能减少预留资源占用。但 Φ_{base} 仍不能满足需要，大量普通任务的等待时间增加，如图 2(c)所示，执行成功率下降，如图 2(e)所示。在这个阶段，VIP 任务的执行未受影响。在接近 20 min 时，系统再次趋于平稳状态。

第 20~30 min 为第 3 阶段。在第 21 min，VIP 任务出现激增，如图 2(a)所示，预留资源池 Φ_{resv} 出现资源不足，滑动窗口 ω 迅速升为最大值 0.6，如图 2(b)所示。大量的 VIP 任务导致了任务等待时间的少量增加，如图 2(c)所示，但 VIP 任务的执行成功率基本没有变化。在这个阶段，虽然普通任务在数量上没有变化，但滑动窗口调整导致普通资源池 Φ_{base} 变小，因此仍然对普通任务的等待时间、执行成功率造成影响，分别如图 2(c)、图 2(e)所示。

第 30~45 min 为第 4 阶段。在这个阶段，普通任务与 VIP 任务先后出现了增长，并持续了较长时间，如图 2(a)所示，导致系统中的资源严重不足。为了优先满足 VIP 任务的调度，滑动窗口最终稳定在 0.4 左右，如图 2(b)所示。在这个过程中，VIP 任务的执行基本没有受到影响，但普通任务的等待时间、执行成功率均受到了严重影响，分别如图 2(c)、图 2(e)所示。

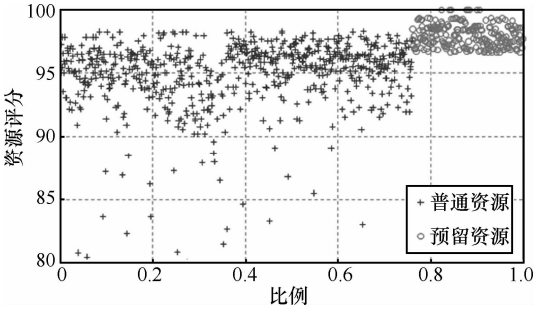
第 45~60 min 为第 5 阶段。随着提交的任务数量回到正常水平，系统又回到平稳状态。

在图 2 展示的整个实验过程中，用户提交的任务数随时间波动很大，SWRR 算法与资源池大小固定的算法相比的优势在于，它通过动态窗口滑动，合理资源预留，始终保证 VIP 任务一个较稳定的任务等待时间与执行成功率。

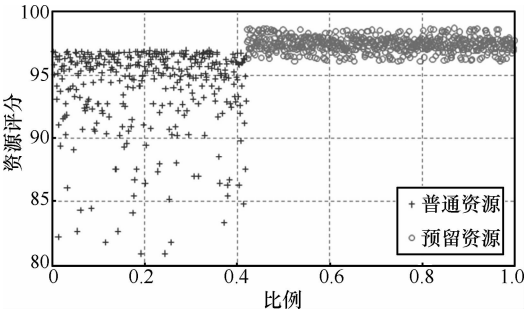
从图 2(d)进一步看出，VIP 任务的执行时间明显小于普通任务。这是因为 SWRR 算法通过资源交换，在预留资源池中维持了最优秀的资源。作为证明，图 3 给出了第 5 min、第 21 min 时的滑动窗口快照。资源评分通过函数资源评价函数 $V(r_i)$得到，它反映了资源在过去一段时间内的任务执行成功率。从图 3 可以看出，SWRR 算法总是将系统中评分最高的那些资源作为预留资源。

3.2　资源负载率

通过预留足够多的资源，一些静态的资源预留算法也能保证 VIP 任务的调度。文献[6]指出一个优秀的资源预留算法不仅要保证 VIP 任务的及时调度，而且要减少资源浪费。图 4 给出了在上述实验过程中，普通资源池与 VIP 资源池的资源负载情况。



(a) 第 5 min 时，窗口大小约为 0.24



(b) 第 21 min 时，窗口大小约为 0.6

图 3　滑动窗口快照

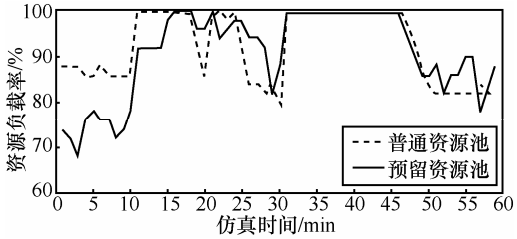


图 4　资源负载率

对比图 2(e)和图 4 可以看出，在资源不足而导致任务执行成功率下降的第 12~15 min、第 22~25 min、第 35~45 min，仅在第 12~15 min 预留资源池的负载率未接近 100%，即存在一定的资源空闲。造成此现象的原因是在第 12 min 时，滑动窗口已达到最小值，如图 2（b）所示，无法进一步缩小以将资源用于普通任务的调度，因此这不是 SWRR 算法本身的缺陷。

在实际部署时，滑动窗口最小值、最大值的设置都需要考虑具体的应用场景。窗口最小值设置过大可能造成资源浪费，设置过小可能瞬间不能适应 VIP 任务的激增；窗口最大值设置过小可能不能为 VIP 任务预留足够的资源，设置过大可能瞬间不能适应普通任务的激增。由于 SWRR 算法可以自适应调整窗口大小（但需要一定的时间），在应用场景不能确定时，可以设置一个较小的最小值、较大的最大值。另外，窗口调整步长、资源交换率实际反映了算法的响应速度与调整粒度，一般来说，对于任务波动较大的场景，可以选择较大的窗口调整步长、资源交换率。

4　结束语

本文提出了一种云计算中的服务可用性保障机制——基于滑动窗口的资源预留算法。通过动态窗口滑动、合理资源预留，SWRR 算法在兼顾所有任务调度的基础上，重点提升 VIP 任务的调度效率与执行成功率，保障重要用户的服务可用性。

SWRR 算法已被应用于一个大型的云计算平台 iVCE 中。接下来的工作主要包括算法执行效率的优化，并对算法在实际系统中的性能进行更深入分析。

参考文献：

[1] SUBASHINI S, KAVITHA V. A survey on security issues in service delivery models of cloud computing[J]. Journal of Network and Computer Applications, 2011, 34(1): 1-11.

[2] CARLIN S, CURRAN K. Cloud computing security[J]. International Journal of Ambient Computing and Intelligence (IJACI), 2011, 3(1): 14-19.

[3] PAWAR C S, WAGH R B. A review of resource allocation policies in cloud computing[J]. World Journal of Science and Technology, 2012, 2(3):165-167.

[4] ENDO P T, DE ALMEIDA PALHARES A V, PEREIRA N N, *et al*. Resource allocation for distributed cloud: concepts and research challenges[J]. Network, IEEE, 2011, 25(4): 42-46.

[5] Amazon elastic compute cloud(EC2)[EB/OL]. http://aws.amazon.com/ec2/.

[6] SEMPOLINSKI P, THAIN D. A comparison and critique of eucalyptus, opennebula and nimbus[A]. IEEE 2nd Int Conf on Cloud Computing Technology and Science[C]. Indianapolis, 2010.417-426.

[7] FUNKE D, BROSIG F, FABER M. Towards truthful resource reservation in cloud computing[A]. IEEE Int Conf on Performance Evaluation Methodologies and Tools[C]. Cargese, 2012.253-262.

作者简介：

沈时军（1982-），男，浙江海盐人，博士，国家计算机网络应急技术处理协调中心工程师，主要研究方向为 P2P 网络、云计算、视频点播。

刘欣然[通信作者]（1971-），男，黑龙江鸡西人，北京邮电大学教授、博士生导师，主要研究方向为分布式计算、信息安全。E-mail: lxr@isc.org.cn。

张鸿（1976-），男，陕西西安人，国家计算机网络应急技术处理协调中心高级工程师，主要研究方向为云计算、信息技术、网络安全。

朱春鸽（1981-），女，河南许昌人，北京邮电大学博士生，主要研究方向为虚拟计算、云计算、网络安全。