$$t_j=1/u_j+1/\lambda_d \tag{11}$$

$$timeExec=\frac{1}{\sum_{j=\frac{numNVP}{2}+1}^{numNVP} C_{numNVP}^{j}R^{j}(1-R)^{numNVP-j}}\cdot\sum_{j=\frac{numNVP}{2}+1}^{numNVP} p_j\cdot t_j \tag{12}$$

$$W_{NVP}=timeExec+1/\lambda_d \tag{13}$$

3.3 检查点容错技术的资源需求、执行开销和响应时间

本文采用最基本的完全检查点容错技术,即周期性地保存云应用系统的关键状态,当系统发生故障时,可恢复到最新的检查点处并继续运行.

- 检查点容错技术的资源需求

检查点容错技术的资源需求量为 1 台虚拟机和 1 个存储单元.虚拟机用来运行检查点程序,该程序主要是周期性地完成检查点工作,存储单元用来存储检查点程序.

- 检查点容错技术的执行开销

根据文献[13],存储开销可用公式(14)表示.虚拟机开销可用公式(15)表示.

$$Cost_{storCP}=\frac{u}{2\sqrt{s}}n_m\cdot 1\cdot T \tag{14}$$

$$Cost_{vmCP}=1\times priceVmHour\times T \tag{15}$$

计算检查点容错技术的软件运行开销时,根据文献[10],首先计算出在时间段[0,T]内检查点的个数,之后用该时间段内检查点的个数与每设置一次检查点所产生的开销的乘积作为其软件运行开销.假设设置检查点需要的时间为 T_{chk},使用一次检查点引起的开销为 $costPerInnovation$,根据文献[6],满足组件 i 在时间段[0,T]内的可靠性需求为 $URelia_i(T)$时的检查点间隔ΔT 可用公式(16)表示.进而,软件运行开销可用公式(17)表示,检查点容错技术的总开销可用公式(18)表示.

$$\Delta T=\left(\frac{T_{chk}}{\ln\frac{1}{URelia_i(T)}}\right)^{0.5} \tag{16}$$

$$Cost_{otherCP}=(T/\Delta T)\times costPerInnovatio \tag{17}$$

$$Cost_{sumCP}=\alpha\times Cost_{storCP}+\beta\times Cost_{vmCP}+\gamma\times Cost_{otherCP} \tag{18}$$

- 检查点容错技术的响应时间

假设所有在检查点和故障期间被执行的事务在恢复期间都将被重新执行,系统可用性为 $avail$,系统执行一个服务需要的时间服从均值为 $1/\mu$的指数分布,故障率为γ',事务请求到达率为λ.那么,根据文献[18],使用检查点容错技术时的响应时间 W_{cp} 可用公式(19)表示.

$$W_{cp}=\frac{\frac{1}{\mu}+avail^2\left(\gamma(\mu\Delta T)^2+\frac{T_{chk}^{\ 2}}{\Delta T}\right)}{avail-\frac{\lambda}{\mu}} \tag{19}$$

其中,$avail=1/(1+\mu\times\Delta T\times\gamma'+T_{chk}/\Delta T)$,$\gamma'=1-URelia_i(T)$.

4 容错技术切换开销的计算

为保证提供商的利益,在生成容错服务时,需要把时间段间容错技术的切换开销考虑在内.包括不同容错技术间的切换,以及都使用复制容错技术但是副本个数不同,或者都使用 NVP 容错技术但是子程序个数不同的情况.假设在时间段[0,T](简记为 T)内使用容错技术 $M1$,在时间段[T,T+1](简记为 $T1$)内使用容错技术 $M2$,那么,在 T 的末期不仅需要保证 $M1$ 的正常运行,还需完成 $M2$ 所需资源的准备,但此时,$M2$ 所需资源未被使用.因此,用准备期间 $M2$ 所用资源的成本开销作为从 $M1$ 容错技术切换到 $M2$ 容错技术的切换开销.

用 $numRepT1$ 表示 $T1$ 内使用复制容错满足容错需求时所需的副本个数,同理,$numNVPT1$ 表示所需的子程序个数.ΔT_{vm} 表示准备一台虚拟机的时间,ΔT_{stor} 表示准备一个存储单元的时间.$perSecdVm$ 表示每台虚拟机每时间单位的开销,并用公式(20)表示,$perSecdStor$ 表示每个存储单元每时间单位的开销,并用公式(21)表示.

$$perSecdVm=priceVmHour/(60\times 60) \tag{20}$$

$$perSecdStor=\frac{\frac{u}{2\sqrt{s}}n_m}{60\times 60} \tag{21}$$

(1) 复制容错技术或 NVP 容错技术切换到检查点容错技术的开销

因为检查点容错技术需要一台虚拟机和一个存储单元的支持,所以,切换开销 $Cost_{toCP}$ 可表示为

$$Cost_{toCP}=perSecdVm\times\Delta T_{vm}+perSecdStor\times\Delta T_{stor}.$$

(2) 检查点容错技术或 NVP 容错技术切换到复制容错技术的开销

因为在 $T1$ 内需要 $numRepT1-1$ 台虚拟机和 $numRepT1-1$ 个存储单元,所以,切换开销 $Cost_{toRep}$ 可表示为

$$Cost_{toRep}=(perSecdVm\times\Delta T_{vm}+perSecdStor\times\Delta T_{stor})\times(numRepT1-1).$$

(3) 检查点容错技术或复制容错技术切换到 NVP 容错技术的开销

因为在 $T1$ 内需要 $numNVPT1-1$ 台虚拟机和 $numNVPT1-1$ 个存储单元,所以,切换开销 $Cost_{toNVP}$ 可表示为

$$Cost_{toNVP}=(perSecdVm\times\Delta T_{vm}+perSecdStor\times\Delta T_{stor})\times(numNVPT1-1).$$

(4) 不同副本个数的复制容错技术间的切换开销

当 $numRepT1>numRep$ 时,切换开销 $Cost_{ReptoRep}$ 可表示为

$$Cost_{ReptoRep}=(perSecdVm\times\Delta T_{vm}+perSecdStor\times\Delta T_{stor})\times(numRepT1-numRep),$$

否则,无切换开销.

(5) 不同子程序个数的 NVP 容错技术间的切换开销

当 $numNVPT1>numNVP$ 时,切换开销 $Cost_{NVPtoNVP}$ 可表示为

$$Cost_{NVPtoNVP}=(perSecdVm\times\Delta T_{vm}+perSecdStor\times\Delta T_{stor})\times(numNVPT1-numNVP),$$

否则,无切换开销.

5 容错即服务提供方案的最优化求解

5.1 容错服务的描述

用下面的多元组表示云容错服务提供商为服务组件 i 在时间段[0,T]内提供的容错即服务.

$$\langle Rep_i(T),NVP_i(T),CP_i(T),vFaultResult_i(T),LRelia_i(T),FRelia_i(T),Para_i(T),costRelia_i(T)\rangle,$$

其中,$Rep_i(T)$、$NVP_i(T)$和 $CP_i(T)$分别表示在 T 内,是否为组件 i 应用复制、NVP 或检查点容错技术,应用时值为 1,否则,值为 0.$vFaultResult_i(T)$表示是否为组件 i 提供了数值容错服务,提供时值为 1,否则值为 0.$LRelia_i(T)$表示组件 i 的可靠性需求是否在提供容错服务时被降低,被降低时值为 1,否则,值为 0.$FRelia_i(T)$表示为组件 i 应用容错即服务时最终达到的可靠性值,即若 $LRelia_i(T)=1$,那么 $FRelia_i(T)=URelia_i(T)-DRelia_i(T)$.$Para_i(T)$表示为组件 i 提供特定容错服务时相应的容错数据参数,如若应用复制容错服务,则给出使用的副本个数;若应用 NVP 容错服务,则给出使用的子程序个数;若应用检查点容错服务,则给出设置的检查点间隔.$costRelia_i(T)$表示在 T 内,当组件 i 被服务的可靠性为 $FRelia_i(T)$时,组件支付给云容错服务提供商的费用.当组件可靠性没有被降低时,使用分配给组件 i 的容错服务带来的总开销(包括容错服务切换开销),作为组件 i 支付给供应商的费用.另外,为了鼓励更多的云应用系统将不重要的组件设置为非关键性组件,以便当容错服务提供商可利用的云资源不足时,可通过降低组件可靠性需求,提供更多的容错服务以提高收益.当组件可靠性被降低时,使用分配给组件 i 的容错服务带来的总开销的一部分,作为组件应该支付给容错服务提供商的费用.本文用 w 表示组件可靠性被降低后,组件支付的费用占总开销的比重.下面以时间段 T 和 $T1$ 为例,分别针对支撑容错服务的底层云资源是否足够的场景,给出可用容错即服务提供方案的最优化求解方法.

5.2 场景1:支撑容错服务的底层云资源足够

在支撑容错服务的底层云资源足够时,使云容错服务提供商总开销最低的容错即服务提供方法就是最优方案.首先给出本节分析所需的一些变量定义.

(1) $rep_T check_{T1}(i)$:若在 T 时间段为组件 i 提供复制容错服务,且在 $T1$ 时间段为组件提供检查点容错服务,则 $rep_T check_{T1}(i)$取值为 1,否则,取值为 0.提供其他容错服务时可同理相关定义.

(2) $Cost_{rep\text{-}cp}(i)$:表示在 T 时间段为组件 i 提供复制容错服务,且在 $T1$ 时间段提供检查点容错服务的成本开销,计算方法为 $Cost_{rep\text{-}cp}(i)=Cost_{sumRep}(T)+Cost_{sumCP}(T1)$.提供其他容错服务时可同理计算这类开销变量.

下文使用最优化方法求解最优的容错即服务提供方案.设置 $Rep_i(T)$、$NVP_i(T)$、$CP_i(T)$的初值均为 1,后续计算根据服务组件的容错需求得到最适合的一种容错服务.设 $k=T$ 或 $T1$.因为云容错服务提供商的虚拟机和存储资源足够,故组件可靠性需求无需被降低,$LRelia_i(T)$和 $LRelia_i(T1)$的值恒为 0,即求解最优容错服务提供方案应满足条件式(22);因为复制容错和检查点容错不能满足数值容错要求,故求解方案应满足条件式(23);因为对于任何组件,如果能够满足该组件对响应时间和数值容错效果的需求,就应为其提供一种容错服务,故求解方案应满足条件式(24);因为所提供的容错技术需满足组件对响应时间的要求,故求解方案应满足条件式(25)~式(27).

$$FRelia_i(k)=URelia_i(k) \tag{22}$$

$$vFault_i(k)\times(Rep_i(k)+CP_i(k))=0 \tag{23}$$

$$(Rep_i(k)+NVP_i(k)+CP_i(k))\times(1-(Rep_i(k)+NVP_i(k)+CP_i(k)))=0 \tag{24}$$

$$(W_{rep}-RT_i(k))\times Rep_i(k)\leqslant 0 \tag{25}$$

$$(W_{NVP}-RT_i(k))\times NVP_i(k)\leqslant 0 \tag{26}$$

$$(W_{cp}-RT_i(k))\times CP_i(k)\leqslant 0 \tag{27}$$

通过上述分析,目标函数定义为云容错服务提供商为所有组件在时间段[0,T]和时间段[T,T+1]内,提供容错服务的总开销最小.该目标函数可用公式(28)描述.

$$\mathrm{MIN}\left(\sum_{i=1}^{p}\left(\sum_{n=1}^{8}x_n\right)\right) \tag{28}$$

其中,p 为请求服务的组件个数,x_n 描述了时间段 T 和 $T1$ 为组件 i 提供容错即服务时,云容错服务提供商的总开销,具体计算方法见表 1.例如,x_1 描述了在 T 时间段为组件 i 提供复制容错服务,且在 $T1$ 时间段提供检查点容错服务时,总开销包括复制容错技术和检查点容错技术本身的资源开销及容错服务切换开销的总和.

Table 1 Total cost of cloud fault tolerance service provider

表 1 云容错服务提供商的总开销

总开销	[0,T]时容错服务	[T,$T1$]时容错服务	开销计算方法
x_1	复制	检查点	$x_1=rep_T check_{T1}(i)\times(Cost_{toCP}+Cost_{rep\text{-}cp}(i))$
x_2	检查点	复制	$x_2=check_T rep_{T1}(i)\times(Cost_{toRep}+Cost_{cp\text{-}rep}(i))$
x_3	检查点	NVP	$x_3=check_T nvp_{T1}(i)\times(Cost_{toNVP}+Cost_{cp\text{-}nvp}(i))$
x_4	NVP	检查点	$x_4=nvp_T check_{T1}(i)\times(Cost_{toCP}+Cost_{nvp\text{-}cp}(i))$
x_5	复制	NVP	$x_5=rep_T nvp_{T1}(i)\times(Cost_{toNVP}+Cost_{rep\text{-}nvp}(i))$
x_6	NVP	复制	$x_6=nvp_T rep_{T1}(i)\times(Cost_{toRep}+Cost_{nvp\text{-}rep}(i))$
x_7	复制	复制	$x_7=rep_T rep_{T1}(i)\times(Cost_{ReptoRep}+Cost_{rep\text{-}rep}(i))$
x_8	NVP	NVP	$x_8=nvp_T nvp_{T1}(i)\times(Cost_{NVPtoNVP}+Cost_{nvp\text{-}nvp}(i))$

这样,满足约束条件式(22)~式(27)并使目标函数最小的解,就是本场景下在时间段 T 和 $T1$ 内,应为云应用系统中各服务组件提供的最优化容错即服务方案.

5.3 场景2:支撑容错服务的底层云资源不足

当云容错服务提供商用于支撑容错服务的底层云资源不足时,某些非关键性组件的可靠性允许被降低,所需的云资源量会有所缓解.此外,云容错服务提供商可外租虚拟机及存储资源,重新生成新的容错服务方案,满足更多组件的容错需求.基于对资源利用灵活性的分析,本文假设云容错服务供应商在为所有可提供容错服务

的组件提供容错服务之前,外包资源引起的成本低于云容错服务供应商因为外包增加的资源使得能提供更好的容错服务而多得的收益.因此,在这种场景下,本文给出的容错即服务提供方法应能保证容错服务提供商在为所有可提供容错服务的组件实施容错服务的条件下,其开销与因为降低组件可靠性而少收取的费用之和最低.首先给出本节分析所需的一些变量定义.

(1) $y_i(T)$:表示组件 i 在 T 内是否被提供容错服务.即若 $Rep_i(k)+NVP_i(k)+CP_i(k)=0$,则 $y_i(T)=0$,否则,$y_i(T)=1$.

(2) $numRep'$表示组件可靠性被降低并应用复制容错技术时需要的副本的个数.$numNVP'$表示组件可靠性被降低并应用 NVP 容错技术时需要的子程序个数.

(3) C_{vm}^0 表示外租一台虚拟机的价格,C_s^0 表示外租一个存储单元的价格.$outV(T)$和 $outS(T)$分别表示在 T 内外租的虚拟机和存储单元的数量.外租资源引起的费用 $outFee(T)=C_{vm}^0\times outV(T)+C_s^0\times outS(T)$.

(4) $preVm_i(T)$和 $postVm_i(T)$分别表示组件 i 在时间段 T 内的可靠性没有被降低和降低后容错所需的虚拟机数量.其中,

$$preVm_i(T)=(numRep-1)\times Rep_i(T)+(numNVP-1)\times NVP_i(T)+1\times CP_i(T);$$

$$postVm_i(T)=(numRep'-1)\times Rep_i(T)+(numNVP'-1)\times NVP_i(T)+1\times CP_i(T).$$

(5) $preStor_i(T)$和 $postStor_i(T)$分别表示组件 i 在时间段 T 内的可靠性没有被降低和降低后容错所需的存储单元数量.通过计算数值可知,$preStor_i(T)=preVm_i(T)$;$postStor_i(T)=postVm_i(T)$.

(6) $sumLoseFee(T)$表示在时间段 T 内,因为降低组件 i 的可靠性而使容错服务提供商少收取的费用,即

$$sumLoseFee(T)=costRelia_i(T)\times(1-w)\times LRelia_i(T)\times y_i(T).$$

(7) $Cost_i(T)$表示在时间段 T 内为组件 i 提供容错服务时的总成本开销,即

$$Cost_i(T)=Rep_i(T)\times Cost_{sumRep}(T)+NVP_i(T)\times Cost_{sumNVP}(T)+CP_i(T)\times Cost_{sumCP}(T).$$

下文使用最优化方法求解最优的容错即服务提供方案.同场景 1 中的设置,$Rep_i(T)$、$NVP_i(T)$、$CP_i(T)$的初值均为 1.设 $k=T$ 或 $T1$.因为只有非关键性组件才能降低其可靠性,故求解最优容错服务提供方案应满足条件式(29);因为支撑容错服务而使用的虚拟机数量不能超过容错服务提供商现有的虚拟机数量(可描述为 $V(T)$)与外租的虚拟机数量之和,支撑容错服务而使用的存储单元数量同理,故求解方案应满足条件式(30)和式(31).此外,求解最优容错服务提供方案时还应满足前述提到的条件式(23)~式(27).

$$LRelia_i(k)+Critic_i(k)\leqslant 1 \tag{29}$$

$$\sum_{i=1}^{p}(preVm_i(k)\cdot C_i(k)+(postVm_i(k)\cdot LRelia_i(k)+preVm_i(k)\cdot(1-LRelia_i(k)))\cdot(1-C_i(k)))\cdot y_i(k)\leqslant v(k)+outV(k) \tag{30}$$

$$\sum_{i=1}^{p}(preStor_i(k)\cdot C_i(k)+(postStor_i(k)\cdot LRelia_i(k)+preStor_i(k)\cdot(1-LRelia_i(k)))\cdot(1-C_i(k)))\cdot y_i(k)\leqslant s(k)+outS(k) \tag{31}$$

通过上述分析,目标函数定义为云容错服务提供商为所有组件在时间段[0,T]和时间段[T,T+1]内,提供容错服务的总开销与因为降低组件可靠性而少收取的费用之和最小.该目标函数可用公式(32)描述.

$$\mathrm{MIN}\left(\sum_{i=1}^{p}\left(\sum_{n=1}^{8}x_n+servedTnonT1_i(T)+servedT1nonT_i(T)\right)+sumLoseFeeTT1\right) \tag{32}$$

其中,$sumLoseFeeTT1$ 是 $sumLoseFee(T)$和 $sumLoseFee(T1)$之和.$servedTnonT1_i(T)$表示仅为组件 i 在时间段 T 内提供容错服务时的开销.同理,$servedT1nonT_i(T1)$表示仅在时间段 $T1$ 内提供容错服务时的开销,即

$$servedTnonT1_i(T)=Cost_i(T)\times y_i(T)\times(1-y_i(T1));$$

$$servedT1nonT_i(T1)=Cost_i(T1)\times y_i(T1)\times(1-y_i(T)).$$

这样,满足约束条件式(23)~式(27)以及式(29)~式(31),并使目标函数最小的解,就是本场景下在时间段 T 和 $T1$ 内,应为云应用系统中各服务组件提供的最优化容错即服务方案.需要特别说明的是,若两个相继的时间段内云容错服务提供商的资源出现不足,如时间段 T 内资源足够而在时间段 $T1$ 内资源不足,那么求解最优方案时使用公式(32)描述目标函数,且 $k=T$ 时满足约束条件式(22)~式(27),$k=T1$ 时满足约束条件式(23)~式(27)及式(29)~式(31).

6 实验及分析

本文使用 CloudSim[19]建立云应用系统运行的云数据中心,使用 Lingo[20]求解最优化容错即服务方案.将本文提出的容错服务提供方法(记为非固定容错服务)与预先设置容错技术的容错服务提供方法(记为固定容错服务),在提供商支付开销、云服务组件支付费用及组件服务质量等方面进行比较.本文将组件个数分别设置为6、9、19 和 29,以模拟不同大小规模的云应用系统.下面分别针对支撑容错服务的底层云资源是否足够的两种实验场景,给出可用容错即服务提供方案的最优化求解过程,并完成本文所提方法的优势分析.

6.1 实验数据准备

不失一般性,参照文献[13,21]完成本实验所需各类数据的设置.将数据中心大小 s 设置为 60TB,数据中心的调节系数 u 设置为 1.设置组件在[0,1]时间段请求容错服务;*URelia* 设置为初始可靠性与(0,0.01)之间的任一随机数的和,初始可靠性由云应用系统研发者初始确定,本文将其设置为(0.935,1)之间的随机数.设置可靠性误差 *DRelia* 为(0,0.01)之间的任一随机数.设置组件响应时间为(0.5,4.5)之间的随机数.表 2 给出了一个具体云应用系统(AID 为 1)的 6 个服务组件在[0,1]时间段的容错需求(注:[1,2]时间段的组件容错需求设置过程相同).

Table 2 Example of fault tolerance requirements of service components in a cloud application

表 2 云应用系统的服务组件容错需求示例

AID_CID	*periodT*	*Critic*	*URelia*	*DRelia*	*RT*	*vFault*
1-1	[0,1]	1	0.996	0	0.681	0
1-2	[0,1]	0	0.966	0.008	3.3	0
1-3	[0,1]	0	0.991	0.003	3.5	1
1-4	[0,1]	1	0.966	0	3.297	0
1-5	[0,1]	0	0.953	0.006	4.0	1
1-6	[0,1]	0	0.958	0.001	3.95	0

设置实验中用到的其他参数的数值.根据亚马逊云平台的资源价格,设置 *priceVmHour* 为 0.085,即每小时每台虚拟机的费用为\$0.085;*perCostStorage* 为 0.15,即 1G 大小的数据每月的存储费用为\$0.15; C_{vm}^0 为\$0.09; C_s^0 为\$0.2;T_{chk} 为 2s;w 为 0.9;权重因子均为 1.设置 *costPerInnovation*=2×(0.085+0.15/(30×24))/(60×60)≈0.00005.设置 λ=6,λ'=0.1,μ=8,c=0.001s,δ=0.00025s,τ_v=0.5s,$\tau_1=1/\mu$,$\tau_j=1/\mu+0.005$,$\Delta T_{vm}=\Delta T_{stor}=1$.

6.2 实验场景1:支撑容错服务的底层云资源足够

云容错服务提供商总开销的实验结果如图 1 所示;组件平均支付费用的实验结果如图 2 所示.

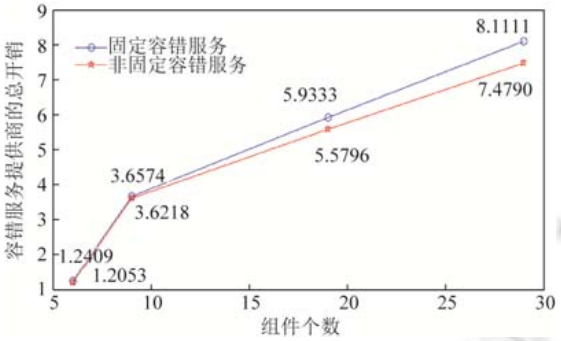

Fig.1 Total cost of fault tolerance provider as resource is enough

图 1 云资源足够时容错服务提供商的总开销

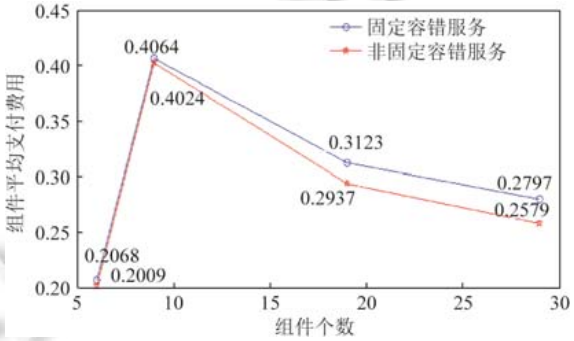

Fig.2 Average expense of components as resource is enough

图 2 云资源足够时组件平均支付费用

分析可知,与固定容错服务模式相比,非固定容错服务提供模式不但降低了云容错服务供应商的总开销,还降低了组件平均支付费用.也就是说,当用于支撑容错服务的云资源足够时,本文所提方法降低了云应用系统需

支付的容错服务费用且云应用系统容错需求得到很好的满足.同时,支撑容错服务的底层云资源的开销有所下降,这样,云容错服务提供商收益更好,提高了容错服务提供商为多个云应用实施高效、可靠容错即服务的能力.

6.3 实验场景2:支撑容错服务的底层云资源不足

组件平均支付费用的实验结果如图 3 所示;云容错服务提供商因为降低组件可靠性而少收取的费用,即降低的容错服务质量的实验结果如图 4 所示;云容错服务提供商的总开销实验结果如图 5 所示.

分析可知,与固定容错服务模式相比,非固定容错服务提供模式不但降低了云容错服务供应商的总开销,还降低了组件平均支付费用,同时很好地保障了云应用系统的容错服务质量.也就是说,当用于支撑容错服务的云资源不足时,本文所提方法降低了云应用系统需支付的容错服务费用且云应用系统容错需求和容错服务质量得到很好的满足.同时,支撑容错服务的底层云资源的开销有所下降,这样,云容错服务提供商收益更好,同样提高了容错服务提供商为多个云应用实施高效、可靠容错即服务的能力.

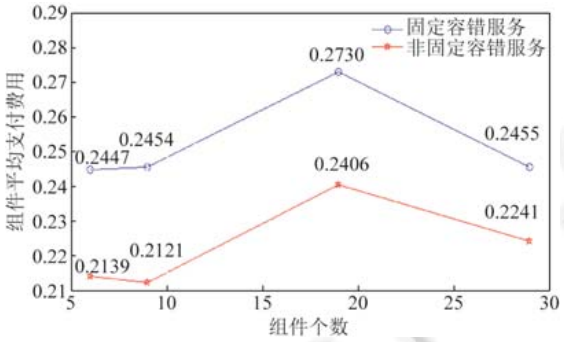

Fig.3 Average expense of components as resource is not enough

图 3 云资源不足时组件平均支付费用

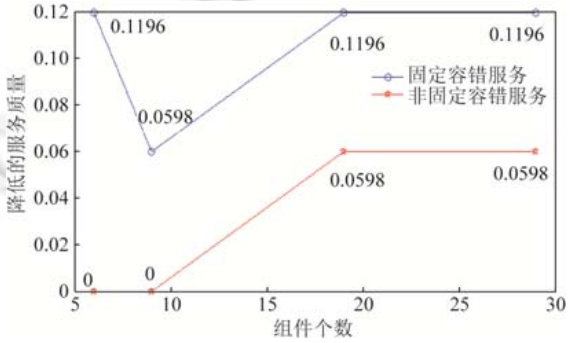

Fig.4 Reduced service quality as resource is not enough

图 4 云资源不足时降低的服务质量

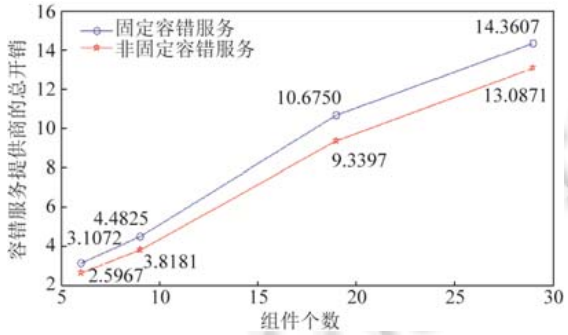

Fig.5 Total cost of fault tolerance provider as resource is not enough

图 5 云资源不足时容错服务提供商的总开销

7 结束语

本文应用云环境下容错即服务的新型模式,提出了一种优化的云容错服务动态提供方法,云应用系统的容错需求重点从服务组件的可靠性、响应时间及数值容错等方面描述,以复制、检查点和 NVP 这 3 种容错技术为基础,分别针对支撑容错服务的底层云资源是否足够的场景,给出可用容错即服务提供方案的最优化求解方法,且容错服务提供方案重点考虑了容错服务动态切换产生的开销.对不同规模的云应用系统容错实际需求进行实验分析,结果表明,与固定容错服务模式相比,从云容错服务提供商实施云容错服务的角度,用于支撑容错

服务的底层云资源的开销有所下降,使服务收益更好,且提高了实施高效、可靠容错即服务的能力;而从云应用系统容错需求的角度,降低了云应用系统需支付的容错服务费用且云应用系统容错需求和容错服务质量得到很好的满足.下一步研究中,一方面将扩展更多容错技术来验证本文所提方法的可用性,另一方面将尝试在真实云应用系统的容错需求环境中确认本文所提方法的实际执行效果.

References:

[1] Jhawar R, Piuri V. Fault tolerance and resilience in cloud computing environments. In: Vacca J, ed. Computer and Information Security Handbook. Elsevier Inc., 2013. 125–141. [doi: 10.1016/B978-0-12-394397-2.00007-6]

[2] Dai HJ, Zhao SL, Zhang JT, Qiu MK, Tao LX. Security enhancement of cloud servers with a redundancy-based fault-tolerant cache structure. Future Generation Computer Systems, 2015,52:147–155. [doi: 10.1016/j.future.2015.03.001]

[3] Wang J, Bao WD, Zhu XM, Yang LT, Xiang Y. FESTAL: Fault-tolerant elastic scheduling algorithm for real-time tasks in virtualized clouds. IEEE Trans. on Computers, 2015,64(9):2545–2558. [doi: 10.1109/TC.2014.2366751]

[4] Jhawar R, Piuri V, Santambrogio M. Fault tolerance management in cloud computing: A system-level perspective. IEEE System Journal, 2013,7(2):288–297. [doi: 10.1109/JSYST.2012.2221934]

[5] Cheraghlou MN, Khadem-Zadeh A, Haghparast M. A survey of fault tolerance architecture in cloud computing. Journal of Network and Computer Applications, 2016,61:81–92. [doi: 10.1016/j.jnca.2015.10.004]

[6] Sun DW, Chang GR, Miao CS, Wang XW. Analyzing, modeling and evaluating dynamic adaptive fault tolerance strategies in cloud computing environment. Journal of Super Computing, 2013,66(1):193–228. [doi: 10.1007/s11227-013-0898-7]

[7] Yi HZ, Wang F, Zuo K, Yang CQ, Du YF, Ma YQ. Asynchronous checkpoint/restart based on memory buffer. Journal of Computer Research and Development, 2014,51(6):1229–1239 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2014.20121125]

[8] Gao Y, Gupta SK, Wang YZ, Pedram M. An energy-aware fault tolerance scheduling framework for soft error resilient cloud computing systems. In: Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE 2014). Dresden: German Press, 2014. 1–6. [doi: 10.7873/DATE.2014.107]

[9] Hamid B, Radermacher A, Vanuxeem P, Lanusse A, Gerard S. A fault-tolerance framework for distributed component systems. In: Proc. of the 34th Euromicro Conf. Software Engineering and Advanced Applications (SEAA 2008). Parma: IEEE Press, 2008. 84–91. [doi: 10.1109/SEAA.2008.50]

[10] Nandi BB, Paul HS, Banerjee A, Ghosh SC. Fault tolerance as a service. In: Proc. of the 6th IEEE Int'l Conf. on Cloud Computing (CLOUD 2013). IEEE Press, 2013. 446–453. [doi: 10.1109/CLOUD.2013.75]

[11] Martin A, Smaneoto T, Dietze T, Brito A, Fetzer C. User-constraint and self-adaptive fault tolerance for event stream processing systems. In: Proc. of the 45th Annual IEEE/IFIP Int'l Conf. on Dependable Systems and Networks (DSN 2015). Brazil: IEEE Press, 2015. 462–473. [doi: 10.1109/DSN.2015.56]

[12] Nakkeeran MM. A survey on task checkpointing and replication based fault tolerance in grid computing. Int'l Research Journal of Engineering and Technology, 2015,2(9):832–838.

[13] Wu XG. Minimum-cost based data replication strategy in cloud computing environment. Computer Science, 2014,41(10):154–159 (in Chinese with English absract). [doi: 10.11896/j.issn.1002-137X.2014.10.035]

[14] Al-Karaki JN. Performance analysis of repairable cluster of workstations. In: Proc. of the 18th Int'l Parallel and Distributed Processing Symposium (IPDPS 2004). New Mexico: IEEE Press, 2004. 26–30. [doi: 10.1109/IPDPS.2004.1303316]

[15] Yuan S, Guo YB, Liu W. Research on voting algorithm in NMR and NVP system. Application Research of Computers, 2008,25(11): 3463–3467 (in Chinese with English abstract). [doi: 10.3969/j.issn.1001-3695.2008.11.079]

[16] LevitinG. Reliability and performance analysis for fault-tolerant programs consisting of versions with different characteristics. Reliability Engineering and System Safety, 2004,86:75–81. [doi: 10.1016/j.ress.2004.01.002]

[17] Imamura K, Heckendorn RB, Soule T, Foster JA. *N*-version genetic programming via fault masking. In: Proc. of the 5th European Conf. on Genetic Programming. Kinsale: Springer-Verlag, 2002. 172–181. [doi: 10.1007/3-540-45984-7_17]

[18] Wolter K. Stochastic Models for Fault Tolerance: Restart, Rejuvenation and Checkpointing. New York: Springer-Verlag. 2010. 1–20. [doi: 10.1007/978-3-642-11257-7]

[19] Calheiros RN, Ranjan R, Beloglazov A, Rose C, Buyya R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software Practice and Experience, 2011,41(1):23–50. [doi: 10.1002/spe.995]

[20] Lingo (home page). http://www.lingo.com/

[21] Zheng ZB, Lyu MR. Fault tolerance management in cloud computing: Selecting an optimal fault tolerance strategy for reliable service-oriented system with local and global constraints. IEEE Trans. on Computers, 2015,64(1):219–232. [doi: 10.1109/TC.2013.189]

附中文参考文献:

[7] 易会战,王锋,左克,杨灿群,杜云飞,马亚青.基于内存缓存的异步检查点容错技术.计算机研究与发展,2014,51(6):1229−1239. [doi: 10.7544/issn1000-1239.2014.20121125]

[13] 吴修国.云计算环境下面向最小成本的数据副本策略.计算机科学,2014,41(10):154−159. [doi: 10.11896/j.issn.1002-137X.2014.10.035]

[15] 袁顺,郭渊博,刘伟.NMR 及 NVP 系统中表决算法分析与研究.计算机应用研究,2008,25(11):3463−3467. [doi: 10.3969/j.issn.1001-3695.2008.11.079]

杨娜(1991－),女,山东临沂人,硕士,主要研究领域为软件容错.

刘靖(1981－),男,博士,副教授,CCF 高级会员,主要研究领域为云计算,容错计算,软件测试.