

基于服务部署的高可用模型及其可用性分配算法

汤海鹰^{1),2)} 许 鲁¹⁾

¹⁾(中国科学院计算技术研究所 北京 100080)

²⁾(中国科学院研究生院 北京 100039)

摘 要 传统高可用系统存在可扩展性较差的问题,文中提出基于服务部署的高可用系统模型,利用对计算资源与存储资源的分离管理实现虚拟高可用服务节点,通过不同服务间共享冗余资源提高资源利用率,有效解决可扩展性问题.基于服务部署高可用系统的关键问题为如何根据服务可用性期望值和使用模式等合理分配资源.针对本系统可用性分配特点,文中提出了最适合冗余优先分配算法,基于结合费用与惩罚值的目标函数得到满足需求的相对最优解,实验证明此算法能较好地达到系统的实时服务部署要求.

关键词 服务部署系统;高可用系统;可用性评估;资源利用评价;可用性分配

中图法分类号 TP302

High-Availability Model Based-on SonD and Its Availability Allocation Algorithms

TANG Hai-Ying^{1),2)} XU Lu¹⁾

¹⁾(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

²⁾(Graduate University of Chinese Academy of Sciences, Beijing 100039)

Abstract Limited scalability is a common problem for traditional high availability systems. With management techniques of Blue Whale SonD system, a High-Availability model based-on SonD is proposed in this paper. This model has better scalability for system management and resource utilization comparing with traditional high availability system models. With the dynamic mapping mechanism between computing resources and storage resources provided by Blue Whale SonD system, this model implements a kind of virtual high availability server by separated management of computing resources and storage resources for various services, releasing the complexity of management and deployment. Since redundant resources could be shared by different services, this model also improves the resource utilization ratio. The key issue of High-Availability system based-on SonD is how to allocate server resources for various services to satisfy the different availability requirements and runtime modes of each service. This availability allocation problem is different from traditional redundancy allocation problems on allocation pattern. According to the specific availability allocation pattern of this system, a Best-Fit Redundancy-Prior allocation algorithm is proposed, which aims to find better solution when considering values of both cost and penalty in objective function. The experiments show that this algorithm achieves the allocation requirement of real time service deployment practically.

Keywords SonD system; high availability system; availability evaluation; resource utilization evaluation; availability allocation

1 引 言

相关数据显示,服务失效将给服务提供商造成每小时上百万美元的巨大经济损失,同时带来顾客满意度降低、服务商信誉下降等严重隐性损失^[1].专用容错计算机虽然能够提供更高的可靠性,但其购买、使用和维护的费用比较昂贵,和开放系统相比缺乏灵活性.基于这种情况,具有更高性价比、以分布式和开放系统为基础的高可用系统逐渐被业界采纳.经过二十多年的研究和发展,高可用系统越来越广泛应用于高性能计算、银行、电子商务等对持续服务能力要求较高的各个领域.

现有大部分高可用系统由于采用共享直连存储,支持节点数量由共享直连存储的最大接口数量所限制,大部分系统中节点小于 32 个,其所支持服务种类同样也有一定限制.这在一个服务节点数量有限的系统中是可行的,但在一个为多个用户提供不同服务、节点数量达到百级甚至千级类似数据中心系统中,由于系统需要分割为多个子系统来实现可用性,这带来了 3 个问题:

(1) 需要为每个子系统部署可用性系统,增加了系统部署的工作量和复杂性;

(2) 增加了系统管理的复杂度.系统管理员需要对每个可用性服务子系统进行单独管理的工作量在一个包含几十甚至上百个子系统的系统中是难以想象的;

(3) 系统资源利用率较低.可用性系统一般都采用资源冗余方式实现,不同可用性子系统之间无法共享资源造成一定程度的系统资源浪费.

从使用和管理角度看,我们需要一个更简单的系统,在保证服务可用性的同时能够简化服务部署和管理;从资源利用的角度看,我们需要一个有效分配各类资源的系统,最大化地提高资源利用率.

蓝鲸 SonD 服务部署系统^[2]是一个基于网络存储的新型计算环境,通过将计算资源和存储资源动态映射成虚拟服务器,提高了管理和系统部署的灵活性,将资源管理简单化,集中管理计算资源和存储资源,提高了对计算资源和存储资源的有效利用率.网络存储使用在高可用系统中从根本上消除了对节点数量的限制,网络存储技术为高可用系统的可扩展性提供了底层支持.当前存储行业已经实现了冗余、备份等多项技术来保证存储设备的高可用性,在此基础上可以实现存储资源的高可用.同时利用对

系统中的计算资源和存储资源的分别管理来为用户提供高可用性的服务器节点,消除系统中存在的单一失效点,则在此系统中部署的服务也能够保证其可用性,不需要针对某种应用来设计特定的高可用系统架构.蓝鲸 SonD 服务部署系统的出现给高可用系统提供了一个新的基础架构.基于服务部署系统的高可用系统 HASBS(High Availability System Based on SonD)通过对系统中分离的计算资源和存储资源的可用性管理,为服务提供虚拟高可用服务节点,在不同服务间共享资源,提高了资源利用率.在此系统对多个服务进行部署的过程中,根据各个服务的可用性预期,如何合理分配计算资源和存储资源为服务构建计算环境是一个重要的问题.本文对基于资源利用率的可用性分配方法进行探讨,针对 HASBS 系统的部署特点,提出最适合冗余优先分配算法,满足了 HASBS 系统的实时可用性分配需求.

本文第 2 节描述 HASBS 系统结构;第 3 节对 HASBS 系统中面向服务部署的可用性分配模型进行介绍;第 4 节介绍可用性分配具体算法并给出分析评价;第 5 节介绍相关研究;第 6 节进行总结并给出未来的研究方向.

2 系统概述

2.1 HASBS 系统结构

HASBS 系统是一个基于蓝鲸 SonD 服务部署系统的提供高可用环境的系统.在本系统中,底层资源都可以保证其可用性,所部署的服务不需要构建自己特定的高可用架构. HASBS 系统通过分别对 SonD 系统的 3 个主要组件——管理服务器、计算资源和存储资源进行可用性管理来实现,为用户提供高可用计算环境. HASBS 系统结构如图 1 所示.高可用管理服务器包括提供监控和部署服务的主管理服务器和多个热备从管理服务器,模式为 Active/Passive.高可用存储资源中卷管理服务为 Active/Passive 模式、网络存储服务为 Active/Passive 或者 Active/Active 模式.在网络存储服务为 Active/Passive 模式时,主存储服务器提供卷管理和网络存储服务,其它存储服务器为热备;在网络存储服务为 Active/Active 模式时,主存储服务器提供卷管理和网络存储服务,多个从存储服务器提供网络存储服务,共享同一存储设备的存储服务器互为热备.计算资源的失效切换由管理资源监控管理.当计算资源

失效时,管理资源选择系统中其它的备用计算资源与相应的存储资源建立映射关系,构建一个与失效系统同构的虚拟计算环境——虚拟化高可用服务节点.在虚拟化高可用服务节点上运行相应服务后,系统可以为用户提供各类应用服务器,如数据库服务器、流媒体服务器和 Web 服务器等.

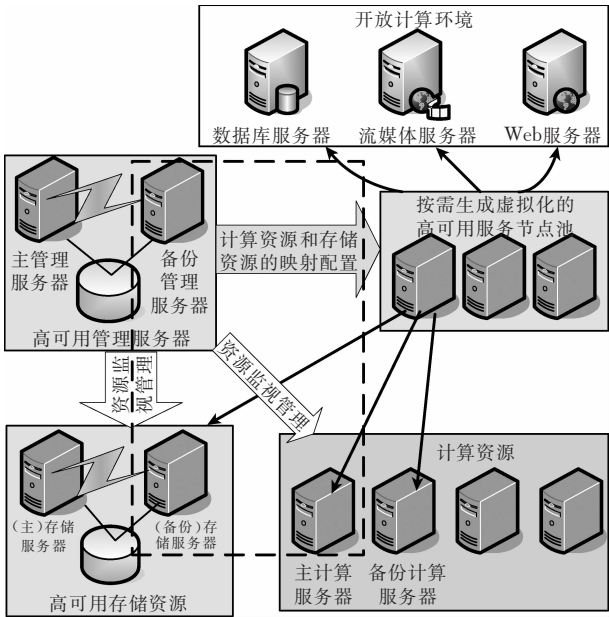


图 1 基于 SonD 的 HASBS 系统架构

(图中虚线框包含的组件为蓝鲸 SonD 部署系统组件, HASBS 系统为各个组件增加了高可用方面的管理,从而消除了蓝鲸 SonD 部署系统的单一失效点,提供了一个虚拟化的高可用服务节点供应用)

2.2 基于存储关联性的服务分类

为了充分利用共享计算资源,我们采用基于存储关联性的分类方式对 HASBS 系统中的服务进行管理.服务属性根据其与底层系统软件的关系可以分为紧耦合(TC)、松耦合(LC)和非耦合(NC)模式. TC 模式的服务和系统软件共存在同一存储系统中,服务只能运行在此特定系统软件之上,此类服务需要独占热备冗余资源,无法与其它服务共享资源;LC 模式的服务使用独立存储系统,但只能运行于某类系统软件上,此类服务可以与其它运行在同类系统软件的服务共享热备冗余资源;NC 模式的服务使用独立存储系统,且可以运行在任何系统软件之上,此类服务可共享冗余资源范围大,可以共享其它所有服务的热备冗余资源.如果冗余计算资源为冷备状态,实际上可以为所有类型服务共享,但热备资源由于失效切换时间较短,在分析可用性时可以忽略失效切换带来的影响,带来更高可用性,所以

HASBS 系统中只包含热备冗余资源. 目前大多数系统中冗余资源均采用热备运行模式.

3 面向服务部署的可用性分配模型

HASBS 系统对多个服务进行可用性部署时存在如何将冗余资源合理分配给各个服务以满足服务的可用性预期且充分利用资源的问题.我们采用结合服务可用性评估和资源利用评价的可用性部署系统来实现系统的冗余分配.面向服务部署的可用性分配模型如图 2 所示.

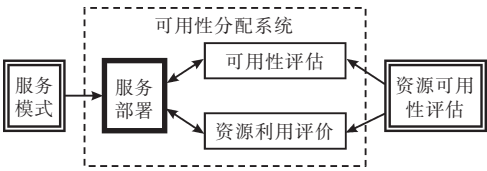


图 2 可用性分配模型

本模型中,服务模式与资源可用性评估为输入,服务部署即可用性分配结果为输出.服务模式包括运行的系统结构、耦合模式、预期可用性等.资源可用性评估主要为对系统中各类资源进行状态监控,根据资源的 MTTF 和 MTTR 实测值得到此资源的动态可用性.可用性评估模块根据服务分配和资源可用性得到服务的可用性值,同时将这个值返回给服务部署模块作为参考.资源利用评价模块根据服务分配与资源可用性等对资源的使用情况进行评价,满足资源利用最优的服务分配作为整个系统的输出.本模型中较关键的机制为可用性评估机制和资源利用评估机制.

3.1 可用性评估机制

复杂系统的可用性评估一般采用分层模型将可用性评估进行简化^[3]. HASBS 系统中服务、计算资源、存储资源与管理系统的可用性依赖关系如图 3 所示.

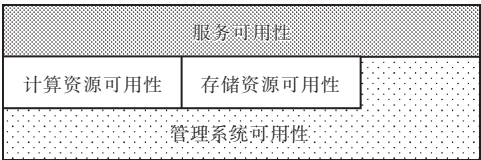


图 3 可用性依赖关系

服务可用性依赖于其它三者的可用性,其中服务的部署直接依赖于管理系统,服务的运行直接依赖于计算资源与存储资源.管理系统负责这两类资源的失效切换过程,则计算资源的可用性和存储资源的可用性直接依赖于管理系统,服务在运行过程

中间接依赖于管理系统的可用性. 这里我们只考虑服务运行过程中的可用性, 即只对服务部署成功后的可用性进行评估. 一种简化的评估方式是对计算资源和存储资源的可用性分别进行评估, 在此基础上对服务的可用性进行评估.

每个服务由计算资源子系统和存储资源子系统以串联模式结合. 一个由 n 个计算资源与 1 个存储资源组成的服务可用性评估简单示例如图 4.

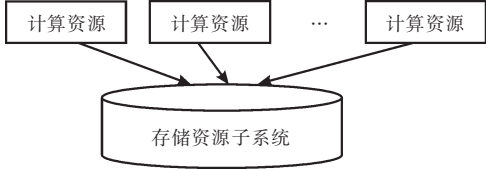


图 4 服务的可用性评估示例

假设只要有一个计算资源可用, 服务可用, 则计算资源子系统的总可用性为 $1 - \prod_{i=1}^n (1 - A_i)$, 其中 A_i 为各个计算资源可用性. 服务可用性为 $A_s = A_d (1 - \prod_{i=1}^n (1 - A_i))$, 其中 A_d 为存储资源子系统可用性. 计算资源子系统比较复杂, 根据服务的运行模式可能是简单并联系统, 也可能是包含平衡负载并联子系统和 k -out-of- n 子系统的串联系统. 目前我们只处理简单并联系统的服务运行模式. 同时本文中不考虑存储资源子系统可用性, 前提条件是存储资源子系统可用性对服务可用性不造成影响, 只有计算资源可用性会改变服务可用性.

3.2 资源利用评价机制

资源利用评价方面主要考虑两个因素: (1) 服务所使用资源总费用; (2) 未满足可用性的服务造成的惩罚值. 两者结合选择相对最优的结果, 即资源总费用和服务惩罚值均较低. 服务资源总费用 C 表示为 $\sum_{i=1}^m C_i$, 服务总惩罚值 P 表示为 $\sum_{k=1}^l P_k \times F(SA_k)$, 其中 C_i 为各服务需要资源费用, P_k 为服务惩罚系数, $F(SA_k)$ 定义如下:

$$\begin{cases} F(SA_k) = 0, & SA_k \geq SA_{0k} \\ F(SA_k) = \frac{SA_{0k} - SA_k}{1 - SA_k}, & SA_k < SA_{0k} \end{cases}$$

SA_k 为服务部署后可用性估值, SA_{0k} 为服务可用性期望值. 公式 $\frac{SA_{0k} - SA_k}{1 - SA_k}$ 被用于可用性性能提升方面的评估^[4], 同样也可以表明可用性性能的相对降低, 因此我们将它借鉴于惩罚值评估中.

基于服务部署的可用性分配与常见冗余分配系

统相比有以下特点: (1) 资源有限; (2) 同时对多个不同模式的服务进行分配; (3) 不同服务间可共享资源; (4) 具有子系统之间资源无关的串-并联系统; (5) 对求解时间要求短以实现服务的实时部署. (1)~(3) 使得本分配系统比以前研究的系统更为复杂. 而对求解时间上的要求决定了本系统的需求不是整个问题的全局最优解, 而是需要所用算法必须能够在较短时间内得到一个相对最优解.

4 可用性分配算法及评价

HASBS 系统可用性分配算法的目标函数中, 我们综合考虑费用 (cost) 和惩罚值 (penalty), 目标函数定义为 $\min(a \times C + b \times P)$, 其中 C 为总费用, P 为总惩罚值, a 和 b 分别为两者的权重. a 和 b 的取值一般通过经验得到, 这里我们将两者均取值为 0.5, 即 cost 和 penalty 比重相同, 目标函数为 $\min(0.5 \times C + 0.5 \times P)$. 在实际应用中可以根据经验或运行要求选择相应值.

HASBS 系统可用性分配问题的搜索空间与系统中的服务数量、类型、服务器数量和冗余资源数量有关. 假定服务数量为 N_{se} , 其中 TC 服务数量为 N_{setc} , LC 服务数量为 N_{selc} , LC 服务类型数量为 N_{lc} , NC 服务数量为 N_{senc} , 服务所需的冗余资源数量均为 N_r , 那么部署所有服务需要的服务器总数量为

$$\begin{aligned} N_{srnum} &= (N_r + 1) \times N_{setc} + \sum_{i=1}^{N_{lc}} (N_r + N_{selci}) + N_{senc} \\ &= (N_r + 1) \times N_{setc} + N_{selc} + N_r \times N_{lc} + N_{senc} \\ &= N_{se} + N_r \times (N_{setc} + N_{lc}), \end{aligned}$$

已知服务器总数量 N_{sr} 及服务所需服务器数量 N_{srnum} , 可用性分配问题的搜索空间为

$$N_{sr} \times (N_{sr} - 1) \times \cdots \times (N_{sr} - N_{srnum} + 1),$$

其中 $N_{srnum} \geq 2$. 例如在 100 台服务器节点的系统中部署 20 个服务, 其中包括 2 个 TC 服务、2 个 NC 服务、16 个属于 5 类不同 LC 模式的服务, 每个服务的冗余资源数量为 1, 部署这些服务所需要的服务器数量为 $20 + 1 \times (2 + 5) = 27$, 则可用性分配问题的搜索空间为 $100 \times 99 \times \cdots \times 74 \approx 2.088 \times 10^{52}$. 由于 HASBS 系统需要在尽量短的时间内得到更优解, 所以获得一个较优初始解比搜索算法更重要.

4.1 最适合分配算法

为了获得较好初始解, 我们提出了一个基本分配策略: 最适合 BF (Best-Fit) 分配策略. BF 策略基于文献[5]提出的定理: 可靠性分配算法中并-串联

系统获得最优可靠性分配方式的重要条件是并联子系统内部的冗余组件完全相同。

目前较通用方式是使用 9 的个数来表示服务可用性等级,根据 9 的个数将服务可用性期望值分为 6 个等级,等级 1 表示 0.9,等级 2 表示 0.99,等级 6 表示 0.999999 等.在这个基础上,我们可以得到 BF 策略中关键的可用性值范围点 A' ,对不同可用性 A 满足 $A=1-(1-A')^2$,如表 1.

表 1 BF 可用性值分配点	
可用性等级	BF 可用性值
1	$g_1=0.6838$
2	$g_2=0.9000$
3	$g_3=0.9684$
4	$g_4=0.9900$
5	$g_5=0.9968$
6	$g_6=0.9990$

表 1 中的第 2 列可用性值表示为满足第 1 列可用性等级所需要的最适合服务器的可用性值.服务器可以根据此可用性值分配点分为 6 类:

- (1) Sg_1 :此类服务器可用性值范围为 $[g_1, g_2)$;
- (2) Sg_2 :此类服务器可用性值范围为 $[g_2, g_3)$;
- (3) Sg_3 :此类服务器可用性值范围为 $[g_3, g_4)$;
- (4) Sg_4 :此类服务器可用性值范围为 $[g_4, g_5)$;
- (5) Sg_5 :此类服务器可用性值范围为 $[g_5, g_6)$;
- (6) Sg_6 :此类服务器可用性值范围为 $[g_6, 1)$.

根据这个分类我们可以得到不同类别服务器的组合关系表.表 2 描述了为达到特定的服务等级,两个属于相同或不同可用性范围类别的服务器之间的组合关系.例如,为了达到可用性等级为 1 的服务需求, Sg_1 类服务器需要与 Sg_1 类服务器组合,为了达到可用性等级为 2 的服务需求, Sg_1 类服务器需要与 Sg_3 类服务器组合.

表 2 可用性类别服务器组合关系表						
	1	2	3	4	5	6
Sg_1	Sg_1	Sg_3	Sg_5	—	—	—
Sg_2	Sg_1	Sg_2	Sg_4	Sg_6	—	—
Sg_3	Sg_1	Sg_1	Sg_3	Sg_5	—	—
Sg_4	Sg_1	Sg_1	Sg_2	Sg_4	Sg_6	—
Sg_5	Sg_1	Sg_1	Sg_1	Sg_3	Sg_5	—
Sg_6	Sg_1	Sg_1	Sg_1	Sg_2	Sg_4	Sg_6

TC 类服务使用服务器的选择比较简单,最佳分配方式为选择具有其可用性等级所对应可用性分配范围的服务器.LC 类服务由于不同服务共享相同的冗余服务器资源,这些服务的可用性等级可能不同,我们在选择服务器时使用最高最合适策略,即为可用性等级最高的服务选择对应可用性分配范围

的两个服务器,其中一个服务器为共享冗余服务器资源,其它可用性等级的服务根据自身可用性等级和冗余服务器可用性所属范围选择一个可用性组合关系表中相应的服务器.例如,LC 服务包括等级 1,3,4 的 3 个服务,所选择的服务器类型为 Sg_1, Sg_2 和 Sg_4 ,包括两个 Sg_4 类服务器,其中一个 Sg_4 类服务器为共享冗余资源.

在上述基础上,我们提出了 4 类 BF 算法用来产生初始分配解:最适合顺序(Best-Fit Increase, BF-INC)分配算法、最适合逆序(Best-Fit Decrease, BF-DEC)分配算法、最适合冗余优先顺序(Best-Fit Redundancy-Prior Increase, BF-RP-INC)分配算法和最合适冗余优先逆序(Best-Fit Redundancy-Prior Decrease, BF-RP-DEC)分配算法.

BF-INC 分配算法的具体流程如下:

1. 按可用性值和费用值递增顺序对服务器进行排列,得到未分配服务器集合 $SetSr$,即对于 $SetSr=\{Sr_1, Sr_2, \dots, Sr_n\}$,满足 $Asr_1 \leq Asr_2 \leq \dots \leq Asr_n$,若 $Asr_i = Asr_{i+1}$,则 $Csr_i \leq Csr_{i+1}$;
2. 将服务按照耦合类型组合,根据各类型期望的最大可用性值和惩罚值递增顺序排列,即对于 $\{St_1, St_2, \dots, St_m\}$ (St_i 为相同耦合类型的服务集合),满足 $Max_Ast_1 \leq Max_Ast_2 \leq \dots \leq Max_Ast_n$,若 $Max_Ast_i = Max_Ast_{i+1}$,则 $Pst_i \leq Pst_{i+1}$;
3. 按照 St_i 到 St_m 的顺序运行步 4~6;
4. 根据表 1 和 Max_Ast_i 得到可用性类别 $Sg_j (j \in \{1, \dots, 6\})$;
5. 选择属于 Sg_j 类的未分配服务器 Sr_k 作为冗余服务器,若不存在,则选择更高类别的未分配服务器,若不存在,则选择 $SetSr$ 集合中可用性值最高的服务器, $SetSr = SetSr - \{Sr_k\}$;
6. 对 $Se \in St_i$,根据表 2 和 Asr_k 得到服务器可用性类别 $Sg_{j'} (j' \in \{1, \dots, 6\})$,选择属于 $Sg_{j'}$ 类的未分配服务器 Sr_l 作为冗余服务器,若不存在,则选择更高类别的未分配服务器,若不存在,则选择 $SetSr$ 集合中可用性值最高的服务器, $SetSr = SetSr - \{Sr_l\}$;
7. 所有服务分配完毕,结束.

其它 3 种分配算法与 BF-INC 其它分配算法的不同点如下:

- (i) BF-DEC 算法从可用性值最高的服务类型开始分配,即第 3 步为按照 St_m 到 St_1 的顺序分配;
- (ii) BF-RP-INC 算法优先分配所有的冗余服务器资源,然后分配其它服务器资源,分配顺序从可用性值最低的服务类型开始,即运行步骤为 $3 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 6$;
- (iii) BF-RP-DEC 算法同样优先分配所有的冗

余服务器资源,然后分配其它服务器资源,但是分配顺序从可用性值最高的服务类型开始,运行步骤与BF-RP-INC类似,只是第3步按照 St_m 到 St_1 的顺序分配.

产生初始解后,本系统实时服务部署要求求解时间复杂度尽可能小,我们没有使用如启发式算法等复杂搜索优化算法,而是采用了两步简单优化策略:

(1) 对于惩罚值不为 0 的服务,搜索使用相同等级服务器节点的其它服务,互换所使用服务器,选择相对优解;

(2) 对惩罚值为 0 的服务,搜索未分配的服务器节点,是否存在使得惩罚值为 0 而费用更低的服务器节点.

4.2 仿真实验

为了简化实验同时能够充分阐述问题,我们在实现时采用以下假定条件和参数赋值准则:

- (1) $N_r=1$.
每个服务有且只有一个冗余资源;
- (2) $N_{senc}=0$.
由于 NC 服务对部署影响很小,假定系统中不存在 NC 服务;

(3) $N_{setc}=\left\lfloor \frac{N_{se}}{10} \right\rfloor$.

- TC 服务占有所有服务的 10%;
- (4) $N_{lc}=5$.
LC 服务类型数量为 5;
- (5) 服务可用性 A_{se} 为取值范围为 $[1,6]$ 的整数;

(6) $P_k=100\times A_{se}$.
其中 P_k 取值随机范围为 $\pm 10\%$;

(7) 服务器可用性 A_{sr} 取值范围为 $[0.70,0.9999]$.
由于可用性高于 0.9999 的服务器性价比较低,本系统不予考虑.文献[6]对一个包含 512 节点的 White 集群系统运行 4 年的实际记录和测量结果进行分析,测得服务器节点的平均可用性为 0.9872,其中大部分服务器节点的可用性大于 0.95,小部分可用性小于 0.8.据此实际环境中的运行结果我们将服务器可用性的分布比例设置如表 3.

表 3 服务器可用性比例

范围	比例/%
$[0.7,0.8)$	10
$[0.8,0.95)$	20
$[0.95,0.999)$	60
$[0.999,0.9999)$	10

(8) 根据文献[7]中可用性与费用的非线性关系,我们定义服务器的费用 C_{sr} 与可用性 A_{sr} 的关系如下:

$$\begin{cases} C_{sr}=A_{sr}^2, & A_{sr}\leq 0.99 \\ C_{sr}=(98.01+((A_{sr}-0.99)\times 10^3)^2), & 0.99<A_{sr}\leq 0.999 \\ C_{sr}=(179.01+((A_{sr}-0.999)\times 10^4)^{2.5}), & 0.999<A_{sr}\leq 0.9999 \end{cases}$$

其中 C_{sr} 取值的随机范围为 $\pm 10\%$.图 5 为本实验所使用的一个包含 400 个服务器节点实例的服务器可用性与费用关系图,可以看出较好地满足了可用性与费用的非线性关系.

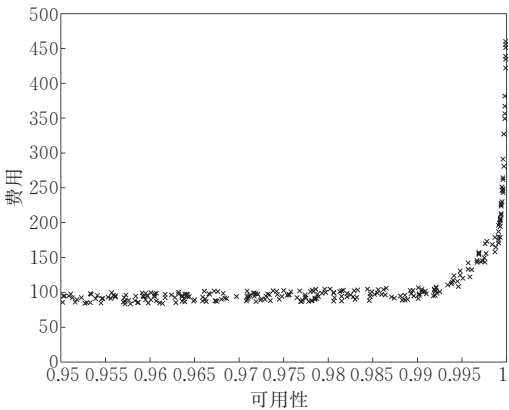


图 5 服务器的可用性与费用关系图

4.3 结果与分析

在 BF 分配算法中,服务器根据可用性值分为 6 类,每次总是从每类最低可用性值的服务器开始分配,可知 BF 分配算法分配准则是分配各类服务器中的最低可用性值部分.4 种 BF 分配算法选择服务器准则相同,不同点在于服务顺序.当每类服务器数量能够满足服务需求时,4 种算法虽然分配方案不同,但是由于分配时选择服务器相同可知总费用相同,而服务可用性期望能够得到满足可知惩罚值为 0,则由总费用和总惩罚值组成的目标函数值均相同,即 4 种算法的解相等.当存在某类服务器无法满足服务需求时,4 种算法得到的解将出现差别.

顺序分配算法与逆序分配算法比较

当高可用性服务器数量无法满足需求时,逆序分配算法中可用性等级高的服务优先分配到所需服务器可用性等级低的服务器,未分配到所需服务器,顺序分配算法则相反,可能导致顺序分配算法与逆序分配算法的两种不同结果:

(1) 如果服务器数量可以满足部分高可用性等级服务,由于可用性等级高的服务 P_k 和 $F(SA_k)$ 取

值均大于可用性等级低的服务,可知顺序分配算法的惩罚值 $P_k \times F(SA_k)$ 大于逆序分配算法的惩罚值,即逆序分配算法能够得到更优解;

(2) 如果服务器数量无法满足所有高可用性等级服务但是能够满足部分低可用性等级服务,逆序分配算法中所有相应服务的可用性都未能得到满足,顺序分配算法中部分服务的可用性得到满足,可能使得顺序分配算法的惩罚值小于逆序分配算法的惩罚值,即顺序分配算法能够得到更优解。

冗余优先分配算法与非冗余优先分配算法比较

在服务器数量无法满足高可用性等级的服务时,非冗余优化分配算法由于没有优先满足冗余资源的可用性,会提高相同耦合类型的部分服务的可用性等级需求. 据表 2 可知,若 Sg_6 类的冗余服务器资源被 Sg_5 类的冗余服务器资源代替,那么可用性等级为 5 的服务所需服务器由 Sg_4 类提高到 Sg_5 类,加剧了稀缺高可用性服务器的需求,进一步导致系统更严重的不可满足性. 相比之下,冗余优先分配算法能够得到更优解。

我们对不同分配算法产生的初始解进行了比较,图 6 为我们在产生初始解时采用 4 种不同分配方式运行的 50 个实例,每个实例的参数设置满足前部分设置准则,其中总服务器数量为 50,总服务数量为 20,部署服务所需服务器数量为 27. 从图中可以看出,大多数实验中冗余优先分配方式能够比非冗余优先分配方式得到更优的解,这主要是由于决定多个服务可用性的冗余资源可用性比非冗余资源

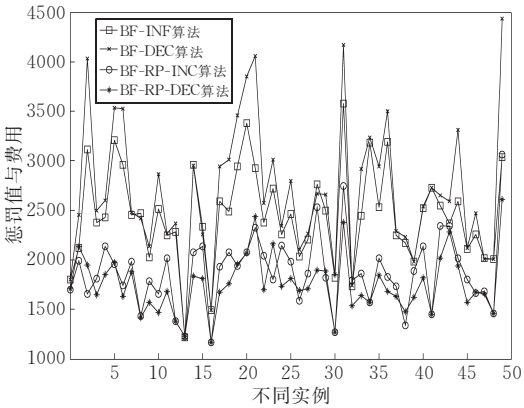


图 6 4 种分配算法初始解比较

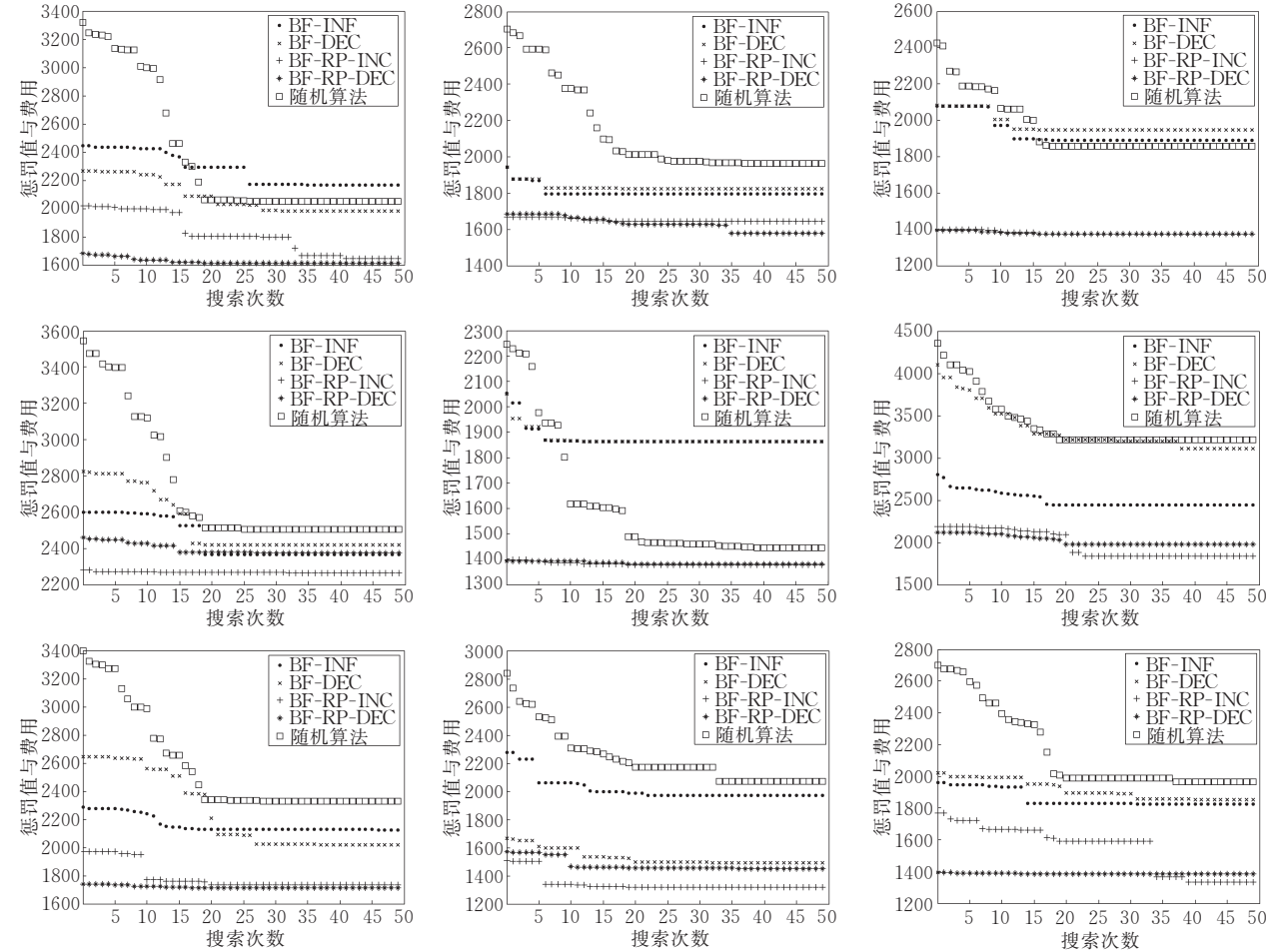


图 7 BF-INC、BF-DEC、BF-RP-INC、BF-RP-DEC 和 Random 算法搜索过程比较

可用性更为重要, 优先满足冗余资源的可用性需求可以满足更多服务的可用性需求. 从实验结果可以看出, 顺序分配方式与逆序分配方式在不同实例中表现不同, 而在实例中最大值都由 BF-DEC 分配方式产生, 从前述算法比较中可知这是由于这些实例中高可用性服务器数量较低, 逆序分配方式中可用性等级高的服务和部分可用性等级次高的服务都无法满足可用性, 从而导致最大惩罚值.

另外, 我们还对以上 4 种分配算法和随机分配算法的搜索优化过程进行了比较. 随机分配算法在分配初始值时不采用任何特定分配准则, 而是在满足共享冗余资源关系的原则上随机地将服务器分配给服务. 这些实验同样也包含 50 个服务器节点和 20 个服务. 图 7 为 9 次不同的实验过程. 可以看出, 由于随机分配方式没有特定分配准则, 在搜索过程中可以比较快速地向最优解收敛, 而其它 4 种分配方式已经采用了较优分配策略, 所以收敛过程较慢, 其中 BF-RP-INC 和 BF-RP-DEC 分配方式的收敛曲线相对更为平滑, 也说明我们的简单搜索算法比较容易陷入局部最优解. 这些实验结果都表明 BF-RP-INC 和 BF-RP-DEC 分配方式相比其它分配方式能够在更短时间内得到更优解, 满足了 HASBS 系统的实时服务可用性部署需求.

5 相关研究比较

高可用技术已经经过多年的研究, 有许多相关系统和产品, 比较典型的高可用系统有双机热备系统 (如 TurboLinux 公司的双机容错系统 TurboHA Server^①)、高可用集群系统 (如 Red Hat 公司提供的集群软件包 Red Hat Cluster Suite^②) 以及多机冷备系统 (openQRM 系统^③). 前两类系统都存在节点数量的限制, 支持的服务有限, 可扩展性较差. openQRM 系统将节点远程启动管理与服务的管理结合, 为用户提供了一个资源统一管理、监控和高可用管理的系统. 它将资源划分为 VE (Virtual Environment), 每个 VE 可能包含多个计算节点资源. 物理节点在第一次网络启动时作为一个空闲资源等待管理系统将它加入一个 VE. 加入一个 VE 后, 节点重新启动加载此 VE 系统镜像. 管理系统监控节点运行状态从空闲资源池中选择计算资源替换失效计算资源提供的服务. openQRM 系统中的冗余资源作为冷备资源, 会在一定程度上降低服务可用性.

冗余分配问题主要包括两类: (1) 满足可靠性

和 weight 的约束条件下, 费用最小; (2) 满足费用和 weight 的约束条件下, 可靠性最大. 传统的冗余分配问题已经被证明为 NP 完全问题^[8]. 相关算法方面的研究取得了比较好的进展, 一些算法, 如启发式算法、遗传算法、模拟退火算法、禁忌搜索算法、神经网络算法、branch-and-bound 算法等都证明了能够比较好地解决这个问题^[9-10], 特别是遗传算法的使用比较广泛. 传统冗余分配算法主要针对一个并串联系统进行分配, 且对分配时间没有特别限定, 最优搜索算法一般需要几十甚至几百秒, 所针对的分配问题类型及需求与我们的系统均有较大差异.

6 结 论

本文给出了基于服务部署的高可用系统模型. 它通过对计算资源和存储资源的分离管理为服务提供了虚拟化高可用服务节点, 同时通过存储关联基础上的服务分类提高了不同服务间共享资源的资源利用率. 在对不同服务进行可用性的实时分配部署时, 最适合冗余优先算法在短时间内得到相对最优解方面有较好性能, 但是在算法的形式理论化分析方面还不够. HASBS 原型系统经过一年多的研发工作已经完成, 可以为应用进行计算资源、存储资源和管理资源的自动失效切换, 切换时间在 10s ~ 30s, 试用结果证明, 能较好地满足当前一些蓝鲸 SonD 系统用户的需求.

在目前工作的基础上还有很多问题有待研究, 对今后的工作我们主要考虑以下几个方面:

(1) 对实际运行环境中资源组件的可用性和服务的惩罚值等的测量和估值, 根据实际环境调整参数和算法;

(2) 对可用性评估进行细化, 根据服务的运行模式采用不同的评估方法;

(3) 对失效切换时产生的重新分配问题进行进一步分析;

(4) 在对部署时间减少限定的情况下, 对搜索算法进行改进, 尽可能靠近全局最优解; 同时由于时间可能较长, 可以在目标函数中增加分配消耗时间因素进行综合考虑.

① Turbo HA Server Technology Whitepaper. <http://www.turbolinux.com.cn/products/data/TurboHA/whitepaper.pdf>, 2001.

② RedHat whitepaper. Delivering High Availability Solutions with Red Hat Cluster Suite. September 2003

③ openQRM, <http://www.openqrm.org/>

参 考 文 献

- [1] Marcus E, Stern H. Blueprints for High Availability. 2nd Edition. Indiana, USA: Wiley, 2003
- [2] Liu Zhen-Jun, Xu Lu, Yin Yang. Blue whale SonD: A service-on-demand management system. Chinese Journal of Computers, 2005, 28(7): 1110-1117(in Chinese)
(刘振军, 许鲁, 尹洋. 蓝鲸 SonD 动态服务部署系统. 计算机学报, 2005, 28(7): 1110-1117)
- [3] Hariri S, Mutlu H. Hierarchical modeling of availability in distributed systems. IEEE Transactions on Software Engineering, 1995, 21(1): 50-58
- [4] Coit D W, Smith A E. Reliability optimization of series-parallel systems using a genetic algorithm. IEEE Transactions on Reliability, 1996, 45(2): 254-260, 266
- [5] Elegbede A O C, Chu Cheng-Bin, Adjallah K H, Yalaoui F.

- Reliability allocation through cost minimization. IEEE Transactions on Reliability, 2003, 52(1): 106-111
- [6] Song H T. Availability modeling and evaluation on high performance cluster computing systems [Ph. D. dissertation]. Louisiana Tech University, Louisiana, USA, 2005
- [7] Hou Wei, Okogbaa O G. Reliability and availability cost design tradeoffs for HA systems//Proceedings of the Reliability and Maintainability Symposium. Virginia, USA, 2005: 433-438
- [8] Chern M S. On the computational complexity of reliability redundancy allocation in a series system. Operations Research Letters, 1992, 11(5): 309-315
- [9] Kuo W, Prasad V R. An annotated overview of system-reliability optimization. IEEE Transactions on Reliability, 2000, 49(2): 176-187
- [10] Gen M, Yun Y S. Soft computing approach for reliability optimization: State-of-the-art survey. Reliability Engineering and System Safety, 2006, 91(9): 1008-1026



TANG Hai-Ying, born in 1977, Ph. D. candidate. Her research interests include network storage and high availability system.

XU Lu, born in 1962, professor, Ph. D. supervisor. His research interests include operating system, computer architecture, and network storage.

Background

Most current high availability systems use direct attached storage model for their storage sharing. The scalability of these systems is limited by this storage model. The availability management and deployment of large scale system would be complex. The resource utilization ratio also would be low due to the separated redundant resources of each subsystem. Blue Whale SonD system uses network storage model, and implements dynamic mapping mechanism between computing resources and storage resources. These techniques enable SonD system to provide a scalable infrastructure for high availability system. This paper proposes a high availability model based on SonD system. In this model, virtual high availability servers, composed of computing resources and storage resources, are allocated to various services. The availability of basic computing resources and storage resources is maintained by management servers of SonD system. The basic redundant resources could be shared by some services. How to allocate the resources to various services to meet their requirement on availability, while increase the resource utilization ratio at the same time, is key to availability deployment system, especially for those with large scale resources. The optimal redundancy allocation problem, or opti-

mal reliability allocation problem, has been proved to be a NP-hard problem. Many heuristic algorithms have been developed to solve this problem. This paper works on the availability allocation problem, which is different from traditional allocation problems in its system structure and allocation patterns. Especially, one important function of the authors' system is to implement real-time deployment of various services. Thus, the runtime of the allocation algorithm is more important than how close the final solution will be to the optimal solution. A good solution which could be obtained in a short time is better than an optimal solution which will be obtained after a long time. A Best-Fit Redundancy-Prior allocation algorithm is proposed to solve the availability allocation problem. The experiments show that this algorithm finds a good initial solution for availability allocation problem

This work is partially supported by the National Basic Research Program (973 Program) of China under grant No. 2004CB318205. This project involves research on service-on-demand deployment model and service quality of network storage. The work in this paper is part of the research on service-on-demand deployment model of network storage, focusing on the reliability and availability of service deployment.