一种主机CPU的可用性模拟方法

王从明 王志坚
(河海大学计算机及信息工程学院 南京 210098)

摘 要 XtremWeb可以汇集联网主机的空闲CPU资源来进行高性能计算,但主机CPU负载是时变的、复杂的和非线性的,具有高度的不确定性。结合ON/OFF模型的思路,提出一个主机CPU的可用性模拟方法,对主机CPU在每日各时刻的可用性进行模拟。实验结果表明,该方法具有较高的模拟精度。

关键词 XtremWeb,主机CPU可用性,重尾分布

Simulation Method for the Availability of Host CPU

WANG Cong-ming WANG Zhi-jian
(College of Computer and Information Engineering, Hohai University, Nanjing 210098, China)

Abstract XtremWeb gathers online idle CPU resources to carry out high-performance computing. However, the host CPU load is time-varying, complex, and non-linear with a high degree of uncertainty. ON/OFF model was introduced to put forward a simulation method for the availability of host CPU for its precise simulation of the process. Experimental results show that the method has higher simulation accuracy.

Keywords XtremWeb, Availability of host CPU, Heavy-tailed distribution

1 引言

HP Laboratories Palo Alto将P2P分为Communication and Collaboration, File Sharing和Distributed Computing 3类[1],SETI@Home, Avaki, Entropia和XtremWeb都属于Distributed Computing。随着SETI@Home的成功,基于P2P的Distributed Computing获得了越来越多的关注。XtremWeb是一个P2P/light weight GRID/Global Computing开源研究项目[2],它基于周期窃取技术,汇集联网的主机空闲CPU资源来进行高性能计算。本文以该项目提供的主机CPU负载样本数据(DEUG 2005)为基础,探讨主机负载的模拟方法,作为评估XtremWeb系统整体计算能力的基础。

主机负载是时变的、复杂的、非线性的,它的显著特点是具有高度的不确定性。Dinda和O'Hallaron在1997年和1998年两次对多台DEC Alpha DUX主机负载进行了长期的采集,获得了大量的数据,总结其特性为[3]:

1)主机负载的变化是一种随机过程;

2)主机负载一般处于较低的水平,但具有很强的波动性;

3)负载值的分布是比较复杂的,呈现符合多种分布的特性;

4)负载随时间的变化具有很强的关联性,过去的负载值对将来有很大影响;

5)负载变化具有高度的自相似性和突发性。

重尾分布是自相似性的一种重要描述方式,可以用来描述诸如分组到达、间隔时间长度、文件传输大小等高可变过程的概率密度,使用重尾分布定义的随机过程更容易进行仿真模拟。ON/OFF模型主要用来模拟具有自相似特性的数据传输过程,本文结合ON/OFF模型,采用重尾分布对具有自相似性的CPU负载进行定义和描述,模拟仿真主机CPU在每日各个时刻的可用情况。

2 重尾分布

定义1 设X为一个随机过程,其累积分布函数cdf为$F(x)=P[X\leqslant x]$,其补函数$ccdf$ $\bar{F}(x)=1-F(x)=P[X>x]$,若$\bar{F}(x)$满足

$$\bar{F}(x)\sim cx^{-\alpha},0<\alpha<2,c>0 \tag{1}$$

则称该随机过程X的密度分布是重尾的,其中α为形状参数。

重尾分布的随机变量,通常具有较高甚至无穷大的方差。Pareto分布是一种比较简单的重尾分布函数,其在所有取值范围内服从幂函数规律,其cdf为:

$$F(x)=P[X\leqslant x]=1-(k/x)^{\alpha},x>k,\alpha>0 \tag{2}$$

其概率分布密度函数pdf为:

$$f(x)=\alpha k^{\alpha}x^{-\alpha-1},x\geqslant k,\alpha>0 \tag{3}$$

并且它的均值为:

$$E[x]=\alpha k/(\alpha-1),\alpha>1 \tag{4}$$

参数k决定了该随机变量x可取的最小值;参数α决定了x的均值和方差,若$\alpha\leqslant 2$,则该分布具有无穷大的方差,若$\alpha\leqslant 1$,则具有无穷大的方差和均值,且α越趋于1,x值的突发性越强。对Pareto和指数分布的pdf在对数坐标尺度上进行比较,指数分布的pdf是一条直线,而Pareto的pdf尾部的衰减要慢得多,这就是"重尾"的含义。

到稿日期:2008-05-05 本文受国家自然科学基金(60573098),国家重点基础研究发展规划(973)(G2002CB312002)资助。
王从明(1975-),男,博士研究生,研究方向为分布式计算,E-mail:wangcm.yss@mailxz.com;王志坚 博士生导师,教授,CCF会员。

目前，有很多方法可以检验一组测量数据是否服从重尾分布，其中，考察 $ccdf$ 在对数坐标尺度上的线性特性来进行判别[4]，是一种比较简单实用的方法。

根据式(1)，可以得到

$$\lim_{x\to\infty}\frac{-d\log\bar{F}(x)}{d\log x}=\alpha \tag{5}$$

由上式可知，在 $x\to\infty$ 时，若数据在对数坐标尺度上趋于一条直线，则可认为服从重尾分布，且该直线的斜率为参数 α 值。

3 主机 CPU 可用性模拟

主机 CPU 的使用率在 0～100%之间快速变化，构成了一个时间序列。一般来说，当使用率较低时，该 CPU 相对空闲，若采用周期窃取技术，就可以将 CPU 资源共享，用来进行分布式计算；当使用率较高时，则不能提供。我们参考 ON/OFF 模型，首先对 CPU 的可用性进行如下定义：

定义 2 令 δ 为一常数，且 $0\leqslant\delta\leqslant100\%$，$p$ 为 CPU 使用率，且 $0\leqslant p\leqslant100\%$。任一时刻 T，若 $p_T\leqslant\delta$，称 T 时刻该 CPU 可用，计为 1；若 $p_T>\delta$，称 T 时刻该 CPU 不可用，计为 0；连续取 1 的时间序列长度称为 on 周期；连续取 0 的时间序列长度称为 off 周期。

根据上述定义，CPU 的可用性可以用一个时间序列表达，该序列在时间轴上表现为 0-1 方波，在 off 周期内全部为 0，on 周期内全部为 1，on/off 周期严格交替出现且独立同分布。其中 on 的周期序列为 CPU 的可用时间序列，是能够共享给分布式计算系统进行计算的时间序列。

一般来讲，可以认为特定主机在每天的同一时刻具有相似的使用情况。我们以此为依据，采集该主机在连续日期同一时刻的 CPU 使用率，采集的数据组成了一个随机过程，我们采用类似定义 2 的方法对该过程进行定义。

定义 3 令 T 为一天内的任一时刻，k 为正整数，若某主机 CPU 在连续 k 天的 T 时刻均可用，则称在连续 k 天的 T 时刻形成了一个 on 周期，计为 π_{on}，且 $\pi_{on}=k$；同样，若连续 k 天的 T 时刻 CPU 均不可用，称在连续 k 天的 T 时刻形成了一个 off 周期，计为 π_{off}，且 $\pi_{off}=k$。

根据定义 3 可知，π_{on}/π_{off} 周期严格交替出现，独立同分布，且 $k\geqslant1$。由于主机在每日的同一时刻具有相似的使用情况，可以初步认为各时刻的 π_{on} 和 π_{off} 序列具有自相似特性，我们采用 Pareto 分布对各时刻的 π_{on} 和 π_{off} 序列分别进行模拟和验证。

根据式(4)，在 T 时刻，π_{on} 和 π_{off} 的均值为：

$$E[\pi_{on}]=k_{on}\alpha_{on}/(\alpha_{on}-1) \tag{6}$$

$$E[\pi_{off}]=k_{off}\alpha_{off}/(\alpha_{off}-1) \tag{7}$$

因为 π_{on} 和 π_{off} 严格交替出现，可以认为时刻 T 的 π_{on} 和 π_{off} 数量近似相等，则 T 时刻 CPU 的可用概率为：

$$P_T=\frac{\sum\pi_{on}}{\sum\pi_{on}+\sum\pi_{off}}\sim\frac{E[\pi_{on}]}{E[\pi_{on}]+E[\pi_{off}]} \tag{8}$$

根据式(5)，利用样本数据，可以得到各时刻的 Pareto 分布参数值，再根据式(6)，式(7)和式(8)，可以得到各时刻的 CPU 的可用概率。

4 实验结果与分析

假设 $\delta=5\%$，即某时刻主机 CPU 使用率若大于 5%，认为该主机 CPU 在该时刻不可用；主机 CPU 使用率若小于 5%，认为该主机 CPU 在该时刻可用。根据提供的 DEUG 样本数据，以每天的第 480 分钟始、第 1080 分钟止，每隔一秒采集一次数据。考虑到数据量和计算量太庞大，我们以分钟为单位，在每天的第 480 分钟～1080 分钟中，每隔 30 分钟抽取一次样本数据，将各采样时刻的 π_{on} 和 π_{off} 周期分别组成两个序列：$\{\pi_{on}(i),i=1,2,\cdots\}$ 和 $\{\pi_{off}(j),j=1,2,\cdots\}$。对 $\forall\pi\in(0,600]$，计 $P_{on}[\pi>\pi_{on}]$，$P_{off}[\pi>\pi_{off}]$ 分别为 $\pi>\pi_{on}$ 和 $\pi>\pi_{off}$ 的概率，以 $\log_{10}(\pi)$ 为横坐标，分别以 $\log_{10}P_{on}[\pi\leqslant\pi_{on}]$ 和 $\log_{10}(P_{off}[\pi\leqslant\pi_{off}])$ 为纵坐标，构成 $\log_{10}$～$\log_{10}$ 坐标系。

数据处理结果表明，各时刻的 π_{on}/π_{off} 序列在 $\log_{10}$～$\log_{10}$ 坐标系上均近似为一条直线(部分结果如图 1 所示)，则认为各时刻的 π_{on} 和 π_{off} 序列均符合截断的 Pareto 分布，根据式(5)，直线斜率即为参数 α 的值，由此得到不同时刻 π_{on} 和 π_{off} 的分布参数 α 的值，如图 2 所示，且在各时刻，$k_{on}=k_{off}=1$。

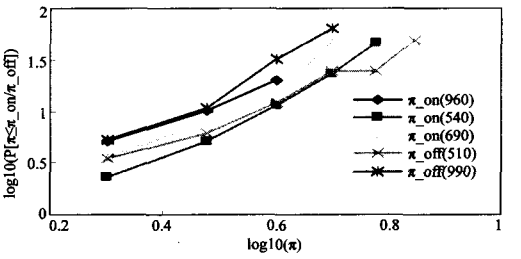

图 1 部分时刻样本数据 Pareto 分布验证

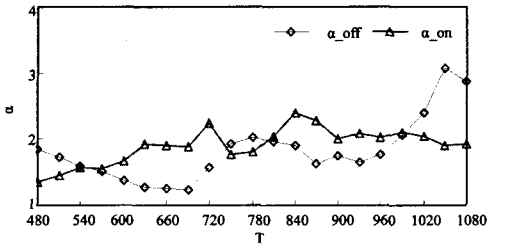

图 2 参数 α_{on}/α_{off} 取值分布图

将不同时刻的参数 k 和 α 的值代入式(6)，式(7)和式(8)，得到主机 CPU 在一日内从第 480 分钟到第 1080 分钟的可用概率分布情况，与实测的概率分布情况进行比较，如图 3 所示，模拟值与实测值误差，如表 1 所列，具有较高的精度。

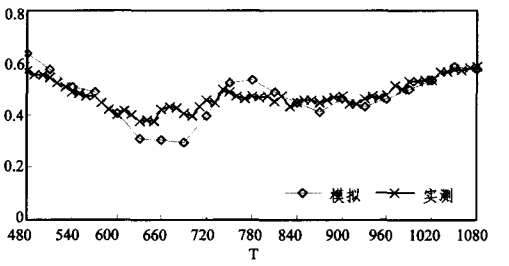

图 3 模拟与实测的 CPU 可用概率比较

由图 2 可知，在第 630 分钟～690 分钟中，α_{off} 值最接近 1，表明该时段 π_{off} 表现出很强的自相似性和突发性，造成模拟值与实测值具有较大的误差，如表 1 所列。

表 1　模拟值与实测值对照表

T	实测值	模拟值	差值	误差
480	0.568966	0.636115	0.067149	11.80%
510	0.543103	0.573380	0.030276	5.57%
540	0.487069	0.506702	0.019633	4.03%
570	0.474138	0.486206	0.012068	2.55%
600	0.400862	0.404685	0.003823	0.95%
630	0.375000	0.308612	0.066388	17.70%
660	0.418103	0.304799	0.113305	27.10%
690	0.405172	0.290681	0.114491	28.26%
720	0.456897	0.395223	0.061673	13.50%
750	0.487069	0.524844	0.037775	7.76%
780	0.469828	0.531866	0.062038	13.20%
810	0.452586	0.489484	0.036898	8.15%
840	0.448276	0.448133	0.000143	0.03%
870	0.443966	0.408159	0.035807	8.07%
900	0.474138	0.460587	0.013551	2.86%
930	0.461207	0.431451	0.029756	6.45%
960	0.478448	0.463714	0.014734	3.08%
990	0.525862	0.497247	0.028615	5.44%
1020	0.534483	0.533204	0.001279	0.24%
1050	0.573276	0.586150	0.012874	2.25%
1080	0.586207	0.574411	0.011796	2.01%

结束语　本文对主机 CPU 在每日各个时刻的可用性进行了较为精细的模拟。实验表明，模拟结果具有较高的精度。根据研究结果，我们将开发主机 CPU 可用性模拟器，来随机产生 CPU 的可用序列，用来对 XtremWeb 系统的整体性能进行评测。

参 考 文 献

[1] Milojicic D S, Kalogeraki V, Lukose R, et al. Peer-to-Peer Computing. HP Laboratories Palo Alto, HPL-2002-57. 2002

[2] Fedak V N G, Germain C, Cappello F. XW: a generic global computing platform-ccgrid'2001 special session global computing on personal devices. IEEE press, 2000

[3] Dirida, DO'Halloran. The statistical properties of host load. Scientific Programming, Also available as CMU Technical Report CMN-CS-TR-98-175. 1999

[4] Crovella M E, Lipsky L. Simulations with Heavy-Tailed Workloads. Self-similar Network Traffic and Performance Evaluation, 2000

(上接第 295 页)

近似解方案总是将电路划分成数量更少面积更大的"块"。这可能是二阶近似解优于 POGA 解的原因。以电路 b21 为例，当 $p=6, r=1.59$ 时二阶近似解划分方案与 POGA 解划分方案分别如图 2 和图 3 所示，其中 C_1 至 C_6 分别表示 6 个替换配置，空白方框表示配置中未使用的区域。

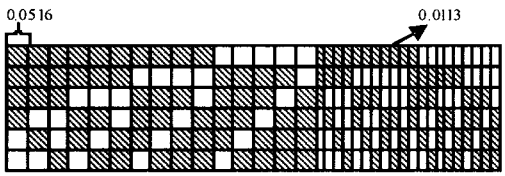

图 3　电路 b21 的 POGA 解划分方案($p=6, r=1.59$)

当 r 较大时($r=2.66$ 或 3.26)，二阶近似解划分方案在同等配置数量下的 MTTF 优于"$m+k$"方案，但并不比 POGA 解划分方案更优。值得注意的是，实验中通过随机搜索也发现了一些 MTTF 略低于 POGA 解，然而在指定的时间段内可靠性 $R(t)$ 高于 POGA 解的方案。这一特点对于某些以故障恢复为主要容错手段的系统而言很重要。

仍以电路 b21 为例，当 $p=6, r=2.66$ 时二阶近似解对应的 $MTTF=5.9723$ 低于 POGA 解。但在 $t=0.5*10^6$ 小时附近二阶近似解方案的可靠性为 0.99820，而 POGA 解仅为 0.99796。

此外，实验表明，在某些情况下将假定 1 的约束适当放宽有助于发现比原近似最优解更优的解。例如当 $p=6, r=1.63$ 时，在放宽假定 1 为 $S(q)+S(q-1)=1$ 时可发现 $MTTF=3.2924/\lambda$ 的划分。此时在原假定 1 下发现的近似最优解仅能达到 $MTTF=3.2863/\lambda$。

结束语　本文研究给定余度和给定替换配置数量的约束下可重构系统的可靠性问题。给出了以可靠性及 MTTF 为优化目标的一阶近似最优化模型及可靠性一阶近似最优化的充要条件；提出了通过二阶近似模型将可重构器件的替换配置资源分配问题转换为二次规划问题进行求解的方法与实施流程。实验表明大多数情况下本方法在可靠性及 MTTF 指标上优于以往的研究。

参 考 文 献

[1] Lach J, Mangione S W H, Potkonjak M. Low overhead fault-tolerant FPGA systems[C]. Very Large Scale Integration (VLSI) Systems, 1998: 212-221

[2] Wei J H. Dependable computing techniques for reconfigurable hardware[D]. UMI Microform 3026836, 2001

[3] Kening Z, Bedette G, Demara R F. Triple Modular Redundancy with Standby (TMRSB) Supporting Dynamic Resource Reconfiguration[C] // Systems Readiness Technology Conference. 2006: 690-696

[4] Emmert J, Stroud C, Skaggs B, et al. Dynamic fault tolerance in FPGAs via partial reconfiguration[C]. Field-Programmable Custom Computing Machines, 2000: 165-174

[5] Gericota M G, Alves G R, Ferreira J M. A self-healing real-time system based on run-time self-reconfiguration[C]. Emerging Technologies and Factory Automation, 2005

[6] Wei J H, McCluskey E J. Column-based Precompiled Configuration Techniques for FPGA Fault Tolerance[C]. Field-Programmable Custom Computing Machines, 2001: 137-146

[7] Subhasish M, Huang W J, Saxena N R, et al. Reconfigurable architecture for autonomous self-repair[C]. Design & Test of Computers, 2004: 228-240

[8] Elshafey K. Embedding fault tolerance via reconfiguration in configurable systems[C]. Microelectronics, 2003: 370-373

[9] Morinaga S. A general model for reliability maximization problem under given redundancy[C]. Fault-Tolerant Computing, 1997: 363-372

[10] Constantinidesz K, Plazaz S, Blomez J, et al. BulletProof: A Defect-Tolerant CMP Switch Architecture[C]. High-Performance Computer Architecture, 2006: 5-16

[11] Basto L. First results of ITC'99 benchmark circuits[C]. Design & Test of Computers, 2000: 54-59