绿色网络存储系统的动力学分析模型

葛雄资[1,3] 冯 丹[1,2] 陆承涛[1] 金 超[1]

(华中科技大学计算机学院 武汉 430074)[1] (武汉光电国家实验室 武汉 430074)[2]

(中国科学院深圳先进技术研究院 深圳 518055)[3]

摘 要 分析和研究了复杂网络存储系统中能耗管理控制的动力学行为规律。通过分析网络存储系统中的磁盘能耗模型,提出一种针对分布式网络存储系统的理想化能耗优化数据布局模型(IEEDP)。在此基础上,结合数据迁移和数据复制技术,提出一种基于二维元胞自动机的绿色网络存储系统模型(GNSSCA)。实验表明,通过节点的局部性调节行为,该系统呈现出复杂的时空演化现象。系统总体副本个数随着负载的增加而出现相应的增加并最终趋于稳定。在负载较低的情况下,节点的访问队列长度熵出现近似的幂律分布。

关键词 网络存储系统,能耗管理,动力学行为,元胞自动机,数据迁移,数据复制

中图法分类号 TP333,TP334 **文献标识码** A

Dynamic Analysis Model of Green Network Storage Systems

GE Xiong-zi[1,3] FENG Dan[1,2] LU Cheng-tao[1] JIN Chao[1]

(School of Computer, Huazhong University of Science and Technology, Wuhan 430074, China)[1]

(Wuhan National Lab for Optoelectronics, Wuhan 430074, China)[2]

(Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China)[3]

Abstract This paper analyzed and studied the dynamic behavior rules on power management and control in complex network storage systems. An Ideal Energy-Efficient Data Placement model (IEEDP) for distributed storage systems was proposed based on the analysis of the disk power model in network storage systems. On the basis of IEEDP, by combining the techniques of data migration and data replication, a 2-D cellular automata model, named Green Network Storage System model (GNSSCA) was proposed. The simulation results show that some complicated temporal and spatial behaviors evolve from the adjustment of local cells. The overall number of replicas of the system increases as the load becomes heavier, and finally it tends to a stable sate. Moreover, when the load is light, it can be seen that there is an approximate power law distribution of the entropy of request queue length.

Keywords Network storage system, Power management, Dynamic behavior, Cellular automata, Data migration, Data replication

1 引言

在全球气候日趋变暖和能源日趋紧张的背景下,节能减排、保护环境已经成为各个行业考量的重要依据点。据调查,2007年数据中心的耗电总量已经达到约200亿度,因耗电而排放的二氧化碳已达到1600万吨,而存储设备则占到数据中心电力消耗的37%~40%。随着数据量的爆炸性增长,存储系统的能耗日益增大。因此,建设更加绿色的海量存储系统显得日益迫切[2]。

网络存储系统通过以太网、光纤等媒介连接成直接连网存储(NAS)、区域存储网络(SAN)或者新出现的结合NAS与SAN的对象存储系统(OBSS)[9]等结构。近年来,随着数据存储的规模越来越大,分布式存储技术得到进一步的发展(例如采用Hadoop架构的云存储和云备份技术[3])。存储节点数目随着应用规模的增大而相应地增长,最终使得网络存储系统发展成为一个非常庞大的系统。面对这样一个复杂的结构,如何从系统的整体行为角度,对网络存储系统的智能性和自组织性等进行分析和研究,具有重要的理论意义和实际应用价值。

传统的单纯依靠马尔科夫链和排队论等理论模型的研究,或者过于强调孤立节点或者单个任务连接等局部性能的优化,从而无法对系统整体行为进行相应管理和控制策略的分析和研究;或者忽略系统的空间连接特性,从而无法分析复杂网络存储系统的时空离散事件,如节点的拥塞、数据流动等。

元胞自动机[10]是一种时间、空间、状态都离散的动力学

到稿日期:2010-09-20 返修日期:2011-01-14 本文受"973"国家重点基础研究发展计划(2011CB302300,2011CB302301),"863"中国高技术研究发展计划项目(2009AA01A401,2009AA01A402),教育部创新团队项目(IRT0725),国家自然科学基金项目(60873028,60933002)资助。

葛雄资(1982—),博士生,主要研究方向为文件系统与存储系统、绿色存储、固态硬盘,E-mail:xiongzige@gmail.com;冯 丹(1970—),女,教授,博士生导师,主要研究方向为文件系统与存储系统、计算机系统结构;陆承涛(1976—),男,博士,主要研究方向为文件系统与存储系统、性能评价、负载特征化;金 超(1983—),男,博士生,主要研究方向为计算机网络存储、存储系统容错。

分析模型，其基本特点是：散布于规则网格中的每一元胞均取有限的离散状态，各个元胞遵循相同的演化规则进行同步演化，各元胞仅和它临近的某些元胞发生相互作用，其状态转移概率根据当前时刻元胞的邻居构型所决定。与传统的模拟方法相比，由于元胞自动机是直接模拟系统的各个成员之间的相互作用过程，从而使得能够通过对构成系统的各局部之间的相互作用的建模，并利用某些合适的转换规则，模拟出极度复杂的演化结果。因此，元胞自动机能够广泛应用于如生物体、流体、地理系统、交通流、网络信息流等复杂系统的演化过程。

人们应用元胞自动机模型和方法对复杂计算机网络中的分组交换以及相变现象进行了仿真研究和理论分析[4-7]。袁坚[5,6]等人通过分析一种计算机网络元胞自动机模型，探讨了网络内部节点的整体行为，结果发现在负载守恒的传输过程中，各网络节点的吞吐量和缓存区的排队长度在空间和时间上均呈现幂率分布，承受系统中局部负载的节点数变化的功率谱呈现出噪声的特点，而网络节点的行为表现出自组织临界现象。

在我们之前的一项研究中[1]，基于复杂网络的二位元胞自动机模型，分别从宏观和微观的角度，探讨了复杂网络存储系统的数据传输与性能的关系。研究表明，整个存储网络发生相变时刻与存储数据对象的产生速度相关，在存储系统中采用数据复制与迁移相结合的机制，能够缩短系统到达负载均衡状态的时间。本文在原有基础上，增加了对存储节点能耗的控制，研究数据本局和数据流动与存储节点能耗和负载均衡的关系，分析和模拟了局部节点数据副本的移动对系统整体能耗与性能的影响。

本文第 2 节中阐述存储系统能耗的相关研究工作，并分析了本文的研究与以往工作的关系；第 3 节中描述和分析了网络存储系统中的磁盘存储能耗模型；第 4 节针对大规模分布式网络存储系统，提出了一种改进的理想化能耗感知数据优化布局模型，并分析了其局限性；第 5 节利用二维元胞自动机进行建模，提出了一种研究绿色网络存储系统的动力学模型；第 6 节中进行实验，并对结果进行了分析；最后总结了全文工作，并对下一步工作进行了展望。

2 存储系统能耗研究

在分布式网络存储系统中，一般包括管理节点和各个数据节点。例如在基于对象的分布式存储系统中[9]，主要包括少量元数据节点和大量数据存储节点，因而网络存储系统的能耗研究主要是对各个数据存储节点的能耗进行管理和控制。一般而言，每个数据存储节点主要能耗部件包括硬盘、CPU、内存等。据统计，在存储系统中，硬盘作为主要的耗能部件之一，总耗能占据整个数据中心的 27%。本文主要探讨的是对于海量硬盘组成的大规模网络存储系统能耗的管理，因此本节主要对磁盘级节能技术进行总结。

对于大规模的网络存储系统的海量硬盘而言，在实际的工作中，大量的硬盘处于空闲状态，或者说同一时刻大量的数据处于非访问状态，因而如何提高系统的能耗利用率是重要的研究问题。对于大规模磁盘系统能耗的研究，主要分为硬件和软件级别的改进。在硬件级别上，目前一个新的研究方向是开发新的低功耗硬件来替代现有的产品，如固态硬盘、混合硬盘等。软件技术上主要分为能耗感知的转移请求技术和数据分布技术等。本文主要从软件级别来进行讨论。

转移请求(Diverted Accesses)技术[11]，利用存储系统中存在的大量冗余数据。利用冗余的数据，读请求到达时，如果数据有某个副本在活跃的磁盘上，并且该硬盘的当前负载可以承担新的请求，则只需要把请求重定向到该磁盘。写请求到达时，如果负载不是非常高，只需要把原始数据写入相应的磁盘，而其它的副本数据或者校验盘数据则会定期写入相应的硬盘。写卸载(Write-off Loading)[14]技术是一种实用的写请求转移技术，当要写的数据磁盘处于低能耗状态时，先将数据写到数据中心其他活跃的磁盘上，因为它的实现是在块一级管理层上，从而对于上层的文件系统和操作系统是透明的，具有较高的使用价值。

能耗感知的数据分布技术主要包括静态的数据布局和动态数据重分布技术。在文献[3]中，作者针对分布式存储系统需要存储大量副本的要求，提出一种能耗感知的静态数据布局方法，使得系统能够提供与能耗成比例的数据带宽服务。该方法忽略了数据请求的不平衡性。本文在第 3 节中，针对数据访问的时空局部性特性，提出了一种改进分析模型。

数据动态重新分布，使 I/O 访问动态地集中在少数几个磁盘上，减少活动的磁盘数量，从而降低存储系统的总体能耗。目前的应用动态数据迁移的低功耗管理的研究中，主要有热点数据集中技术[13](Popular Data Concentration，PDC)、大规模空闲磁盘阵列[12](Massive Array of Idle Disks，MAID)以及可换挡磁盘阵列[8](Gear-Shifting Power-Aware RAID，PARAID)等。PDC[13]技术中，将数据按照访问频率的高低进行分组，将较高访问频率的数据集中存储在几个磁盘上。MAID[12]比较了传统的高性能的磁盘阵列集群存储系统和磁盘库存储系统，提出了一种空闲磁盘海量存储系统，即在一个系统中具有很多休眠磁盘，而不都是活跃状态的，主要是把不经常访问的数据和冗余的数据存储在非活跃硬盘上。MAID 技术最大可为磁盘阵列节省 90%的能耗。Copan、富士通、NEC 和最新的 GreenBytes 等存储软硬件供应商都在其产品应用商中应用了 MAID 技术，其产品有如富士通的 ETERNUS 8000/4000、NEC 的 Storage D 系统、GreenBytes 公司的 GB-X 系列等。PARAID[8]是一种通过分析工作负载进行阵列内部调整的能耗改进方法，即根据需求调整磁盘阵列的活跃硬盘个数，类似于机械操作里面的换挡操作，可以在实际系统中获得很高的节能效应。

以上关于软件级磁盘能耗的研究中，都没有对超大规模的绿色网络存储系统进行建模和分析。本文的研究不同于以上所有的工作，而着重在于利用基于元胞自动机的动力学分析方法，研究大规模网络存储系统中数据访问负载与海量磁盘的能耗以及整体 I/O 性能的关系，并分析网络存储系统的自组织性和自我调节能力，即通过局部数据节点的数据布局和访问的控制，达到全局能耗的优化和性能的平衡。

3 网络存储系统磁盘能耗模型

一般说来，磁盘具有 4 个状态，即活跃(Active)、空闲(Idle)、预备(Standby)和睡眠状态(Sleep)。如图 1 所示，一个磁盘状态周期包括盘片旋转减速过程 T1(Spin-down)和盘片旋转加速过程 T2(spin-up)。在 T1 和 T2 过程中，需要额外的能耗开销 E_{switch}。我们把刚刚能抵消这种额外开销的空闲

周期叫做收支平衡时间 T_{be}(Break-even time)。

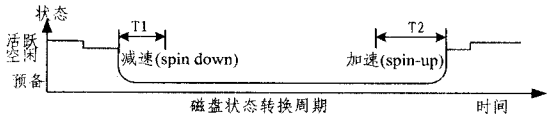


图 1　磁盘状态转换周期示意图

通常在很多实际系统中,硬盘若一直处于空闲状态,且其空闲的时间超过某个阈值,则会自动处于预备状态。在实际应用环境中,则需要较好的预测手段来确保下一段时间内该硬盘是否具有请求到来。提前使硬盘处于活跃状态,能够降低访问延迟,但是可能浪费更多的能量。

大部分情况下,活跃和空闲状态消耗差不多的能量。而睡眠状态下,虽然完全不消费能量,但是需要长时间才能回到正常工作状态。预备状态下,磁盘的磁头和盘片都已经停止移动,但是仍然处于通电状态,能够较快地(10s 左右)回到正常工作状态。在绿色网络存储系统中,为了使系统的响应时间在用户可以承受的范围中不会将磁盘完全关闭,应将其保持在预备状态。因而,在本文的磁盘能耗模型中,我们主要考虑两种状态,即活跃状态和预备状态。

假定在网络存储系统中一共有 n 个节点,每个节点挂载一个大容量硬盘(多个硬盘的情况,考虑一个磁盘阵列的配置),假定第 i 个硬盘经过某种能耗控制软件方法,一共需要经过 K_i 次状态转换过程,该硬盘需要总的能量消耗为:

$$E(i)=E_A(i)+E_s(i)+KiE_{switch}$$

式中,$E_A(i)$和 $E_S(i)$分别表示盘 i 在活跃与预备状态的总功耗,其可以分别计算如下:

$$E_A(i)=T_A(i)P_A$$

$$E_S(i)=T_S(i)P_S$$

式中,$T_A(i)$和 $T_S(i)$分别表示盘 i 处于活跃和预备状态的时间。实际过程中,活跃的时间 $T_S(i)$包含了处于空闲状态的时间以及寻道、旋转和数据传输的时间。分别计算如下:

$$T_A(i)=T_{active}(i)+T_{idle}(i)$$

$$T_{active}(i)=T_{seek}(i)+T_{rotate}(i)+T_{transfer}(i)$$

式中,$T_{active}(i)$和 $T_{idle}(i)$分别表示盘 i 处于活跃和空闲的时间。$T_{active}(i)$、$T_{idle}(i)$和 $T_{idle}(i)$分别表示盘 i 总的寻道、旋转以传输数据的时间。

整个网络存储系统所有磁盘总的能耗计算可以为:

$$E_{total}=\sum_{1}^{n}Ei$$

4　理想化能耗优化数据布局

文献[3]中提出,对于理想的绿色存储系统而言,最优化的管理目标为:当打开的存储节点数目与需要提供的存储性能成正比例时,能较为容易地控制能耗成本。

在分布式存储系统中,假如给定一个大的数据集 DS,如果每个大块(Chunk)的数据大小均为 a,DS 可以划分成 m 个数据块,表示成集合如下:

$$DS=\{C_1,C_2,\cdots,C_m\} \tag{1}$$

理想化能耗优化的数据布局目标是将该数据按照某种策略分布在 n 个磁盘上,每个数据块可以保存一定量的副本,保证提供与能耗成比例的数据读带宽服务(忽略写请求),即每一个节点在打开的时候,都能够提供相同的负载,每一个节点都能得到相同程度的利用。

根据数据块访问的局部性原理,类似于热点数据集中技术(PDC),可以将数据块划分为不同访问频率的组。在这里对数据集中每个块的访问特性进行简单分类,即分为热点和非热点数据,表示如下:

$$DS=\{C_1,C_2,\cdots,C_u,C_v,\cdots,C_m\} \tag{2}$$

式中,热点数据集 $DShot$ 和非热点数据集 $DScold$ 分别表示如下:

$$DS_{hot}=\{C_1,C_2,\cdots,C_u\}$$

$$DS_{cold}=\{C_v,\cdots,C_m\}$$

对于 DS_{hot} 中的每个数据块保存的副本个数为 r_1,对于 DS_{cold} 中的每个数据块保存 r_2。假如非热点数据块几乎不会被访问,则将 DS_{cold} 上的数据块随机分布在存储系统中,只需要考虑热点数据块的分布。设 b_i 为磁盘 i 上存储的热点数据块个数。根据理想化能耗优化的数据布局目标,如果打开了 i 个节点,原始热点数据块数目为 u,则每个数据节点提供相同量的数据服务,于是 $b_i=u/i$。

网络存储系统中数据分块的大小,参照大型分布式系统(如 Hadoop 系统),一般为 64MB 或者更大,因此对于数据块的读请求可以视为顺序的读操作。参照文献[3],设想初始时,尽可能少地在 p 个节点上平均分布热点数据集 $DShot$ 中的 u 个数据块,系统初始时提供的最小性能和能耗配置为同时打开这 p 个节点的情形。

因为打开的磁盘都处于活跃状态,则打开 i 个磁盘的功率 $Power(i)$与打开磁盘个数成正比例。每个活跃磁盘的活跃时间 TA 相同,总的能耗为 $E_{total}=T_AP_Ai$,则总的功率为:$Power(i)=P_Ai$。

打开 i 个节点,吞吐量设为 $Thpt(i)$,于是需要每个配置下提供的吞吐量和能耗功率成相同比例,可以表示如下:

$$\frac{Thpt(p)}{Power(p)}=\frac{Thpt(i)}{Power(i)},\forall i\in\{p,\cdots,n\}$$

设 S 为存储热点数据块的节点个数,则所有存储的热点数据块的总和为:

$$\sum_{i=1}^{s}b_i=u+\sum_{i=p+1}^{s}\frac{u}{i}=ur_1\Rightarrow\sum_{i=p+1}^{s}\frac{1}{i}=r_1-1$$

因 $1/i$ 为单调递减函数,根据上式,其上限和下限分别是

$$\int_{p+1}^{s+1}(1/x)\mathrm{d}x\leqslant\sum_{i=p+1}^{s}1/i\leqslant\int_{p}^{s}(1/x)\mathrm{d}x$$

从而,有 $S\geqslant pe^{r_1-1}$,即所需要存储的磁盘数目随着热点数据副本数目的增加而成指数级增长,意即该方法需要大量的磁盘数目和空间来实现。

预先知道每个数据块的访问频率,并简单地将数据分成热点与非热点两种类型,能够比较好地利用以上的数据分布策略。然而在实际应用中,数据的访问频率却有相当大的差别,若服从 Zipf-like 的分布[16],则需要对于各个级别频率的数据块进行分组。另一方面,数据块的访问频率会随着时间的改变而发生改变,需要动态地控制副本在各个磁盘上的数据分布,从而使得系统的管理更加复杂。如何控制数据的动态分布,并兼顾系统性能与能耗,需要有更进一步的分析模型。

5　绿色网络存储系统动力学模型

5.1　网络存储系统抽象

在分布式网络存储系统中,如分布式对象存储系统中,每

个节点具有自我管理能力，每个数据节点都能够控制自身数据的管理，并且可以跟元数据服务器和其他数据节点进行通信，通过与元数据节点交互之后，客户节点能够与数据节点进行点到点的交互。如图 2 所示，层 1 中表示各个客户节点，层 2 中展示了存储系统中拥有众多的存储节点。

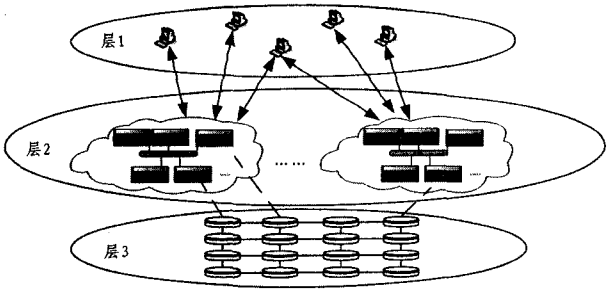


图 2　网络存储系统的存储池抽象模型

对于网络存储的拓扑结构，传统上主要为 Fat-Tree、2 维图、3 维图等结构。元胞空间的 2 维结构能够简单并有效地模拟复杂的网络拓扑结构[4,6]。从图 2 中层 3 抽象出各个数据节点，将网络存储系统抽象为二维格子空间。

在本模型中，假设每个节点下挂载一个同样大小的硬盘或者配置相同的磁盘阵列。每个节点容量大小为 D。系统中存储的数据块集合采用第 4 节中式(1)定义。

数据副本能够提高系统的可靠性，另一方面数据副本的增加能够提高数据的可用性，提高数据访问的负载均衡。而过多数据副本的存在则会增加写操作的负担。每个数据块的副本个数为 r_i，其副本个数上限和下限分别是 $r_{\max}$ 和 $r_{\min}$，如下式所示：

$$r_{\min} \leqslant r_i \leqslant r_{\max}$$

每个节点存储数据块状态用集合 B_i 来表示：

$$B_i = \{Y_{ij} \mid Y_{ij} \in \{0,1\}, 1 \leqslant j \leqslant m\}$$

式中，每个 Y_{ij} 的值代表是否存储了相应的数据块 C_i。

每个节点上数据块总的大小不能超过当前节点最大容量，即 $\sum_{j=1}^{m} aY_{ij} \leqslant D$。

初始时，保证每个块都存储于某个节点上。每个存储点上最多保存原始块对应的 1 个副本，即不会存在有 2 个或多个相同的数据块存储在一个节点上。

5.2　二维 GNSSCA 元胞自动机模型

下面我们定义绿色网络存储系统的元胞自动机模型(GNSSCA)。在 GNSSCA 的二维格子空间 Π 中，每边拥有 N 个数据节点的存储空间，则数据节点总数即为 N^2，其中每个元胞表示一个数据存储节点。GNSSCA 可以利用一个 4 元的组来表示：

$$(\Pi, S, N, F)$$

其中，空间 Π 表示为：

$$\Pi = \{ie1 + je2\}, 0 \leqslant i, j < N$$

每个元胞的邻居关系选择的范围，正是元胞自动机节点之间“局部”相互作用的体现。我们选取 Von-Neumann 型[10]的邻居类型，如图 3 所示，即每个数据节点连接上下左右一共 4 个邻居。模型中的邻居关系可以表示成一个映射：

$$N(X) = \{X+e1, X-e1, X+e2, X-e2\}, X \in \Pi$$

其边界采用周期型。该模型可以表示一种无限的空间拓扑模型，可以用来模拟超大规模的网络存储系统。

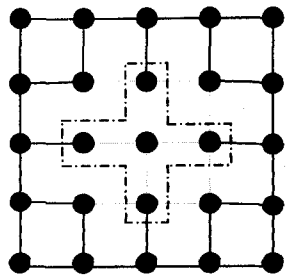


图 3　邻居模型

根据在第 2 节中讨论的磁盘能耗模型，每个节点中的磁盘可以工作在活跃和预备 2 种状态，分别用 1，0 来表示。对于磁盘阵列而言，则表示所有的硬盘都工作在同样状态。每个节点的状态集表示为

$$S \in \{0,1\}$$

$F: Sn \to S$ 表示状态转换规则。

在以下分析中，主要考虑系统中的读请求。

在每个时间步内，每个数据块 C_i 的读请求频率为 $\rho_i(t)$。节点 i 在 t 时刻总的访问频率为 $p_i(t)$，计算如下：

$$p_i(t) = \sum_{j=1}^{m} [\rho_j(t) Y_{ij}]$$

每个节点具有一个读请求队列，其队列长度表示为 $q_i(t)$。当处于活跃状态，每个节点的吞吐量为 $thrpt$。在时间步 t 至下一个时间步 $t+1$ 内最大能够处理 $\frac{thrpt}{a}\Delta t$ 个请求，每个时间步的最大队列长度变化 $\Delta Li(t+1)$ 为：

$$\Delta L_i(t+1) = p_i(t+1) - \frac{thrpt}{a}\Delta t - q_i(t)$$

于是得到下一个时刻 $t+1$ 的队列长度为：

$$q_i(t+1) = \begin{cases} \Delta Li(t+1), & \Delta L_i(t+1) > 0 \\ 0, & \Delta L_i(t+1) \leqslant 0 \end{cases} \tag{3}$$

由此，GNSSCA 模型转移规则 F 如下：

1)每个数据块在一个时间步 t 内访问频率 $\rho_i(t)$。总的访问频率通过式(3)计算可以得到该时刻内的访问队列长度 $q_i(t)$。

2)当队列 $qi(t)$ 的长度大于某个阈值 $l_{\max}$ 时，该元胞将查询其邻居节点的状态。选取处于活跃状态的节点集 C_x，其队列长度处于 $(l_{\min}, l_{\max})$ 之间。如果该节点中含有 $q_i(t)$ 中的副本请求，则转移请求到该邻居节点，并保证邻居节点的请求数目不会超过 $l_{\max}$，直到使得 $q_i(t)$ 的长度小于 $l_{\max}$。如果转移请求无法满足需求，则复制相应的副本至活跃节点，并且该数据块副本总体个数小于 $r_{\max}$。如果以上条件都能消除负载过高，并且邻居节点中有状态为 0 的节点，则将该节点变成活跃状态 1，并转移请求或者复制相应副本至该节点。

3)当等待队列的长度 $q_i(t)$ 小于某个阈值 $l_{\min}$ 时，该元胞将查询其邻居节点的状态。选取处于活跃状态的节点集 C_x，其队列长度处于 $(l_{\min}, l_{\max})$ 之间。如果该节点集中含有 $q_i(t)$ 中的副本请求，则转移该请求到某一个邻居节点，并保证邻居节点的请求数目不会超过 $l_{\max}$。如果 C_x 中的节点都没有该队列的相应数据块，并且有冗余的处理能力，则迁移相应的副本到其中的节点中。如果所有的请求都转移到邻居节点集 C_x 中，则该节点状态由活跃态 1 变为预备状态 0。

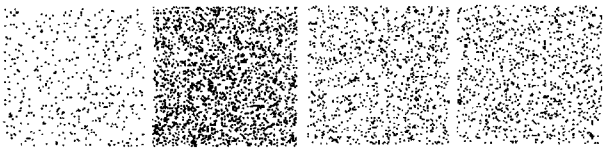
4)循环进入下一个时间步 $t+1$。

6 仿真与结果

设一共有 10000 个数据块，分块大小为 64MB。节点容量为 1TB，即可以最多保存 20 个数据块。节点规模 N 为 100，总节点数目为 10000。每个数据块的最大副本 r_{max} 和 r_{min} 分别为 10 和 1。队列长度阈值 l_{min} 和 l_{max} 分别为 5 和 15。在一个时间步长的读吞吐量 $thrpt$ 设为 64MB/s，即一个数据块。

初始时，随机选取 500 个节点存放原始数据块集合。热点数据的访问服从近似的 zipf 分布[16]，其分布斜率参数 $\beta=\frac{\log\frac{Z_1}{100}}{\log\frac{Z_2}{100}}$，表示 $Z_1/100$ 的访问集中在 $\frac{Z_2}{100}$ 的数据上。在这里用 $\eta=\frac{Z_1}{Z_2}$ 来表示访问的不均衡性。设对每个热点数据块的访问服从达到率 λ 为泊松分布。

初始时设 λ 为 0.2，$\eta=80/20$。节点状态变化如图 4 所示。图 4 中第一代元胞(a)表示初始时的元胞状态图，此时原始数据块随机分布在 500 个节点中，其余节点处于低能耗状态下。经过一段时间(第二代元胞(b))，通过数据块的复制和迁移(转换规则 2 和 3)，大量的节点开始处于活跃状态。第三代元胞(c)和第四代元胞(d)表示处于稳定状态下各个元胞的状态变化，通过长时间的演化，数据分布处于比较理想状况下，各个元胞之间能够通过局部之间的通信，使得大部分的节点处于低能耗状态下，并且总的活跃节点数目处于比较稳定的状态。



(a)第 1 代元胞　(b)第 2 代元胞　(c)第 3 代元胞　(d)第 4 代元胞

(黑色为状态 1，白色为状态 0)

图 4　各元胞状态变化图

当访问负载不同时，系统中的副本个数也会发生变化。如图 5 所示，对不同的负载达到率 λ(分别为 0.2 和 0.5)下系统演化过程中的副本总体个数进行统计。可以看出，当负载增大时，副本总数亦随之增大，并且达到稳定状态所需要的时间也随之增长。

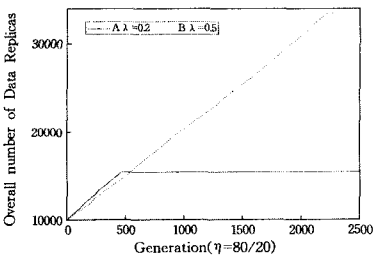


图 5　系统中总的副本数目的变化

通常用熵来衡量元胞自动机的分形和混沌等复杂行为特性[15]。如果用 $L_i(t)$ 表示节点 i 在 t 时刻的队列长度，则节点中含有队列长度为 l 的可能性近似计算为：

$$\phi_l(t)=\frac{1}{N^2}\sum_{i=1}^{N^2}\theta(L_i^t)$$

式中，$\theta(L_i^t)=\begin{cases}1, & L_i^t=l\\ 0, & L_i^t\neq l\end{cases}$。

该时刻的请求队列长度 l_{min} 和 l_{max} 之间的熵为：

$$H(t)=-\sum_{l_{min}\leqslant l\leqslant l_{max}}[\phi_l(t)\log\phi_l(t)]$$

如图 6 所示，当访问的达到率分别为 0.2，0.5 时，请求队列长度熵服从近似的幂率分布。当负载较高(访问达到率为 1.5)，且访问的局部性不明显(80∶80)，队列长度熵最终趋于 0，整个系统的吞吐量达到极限。

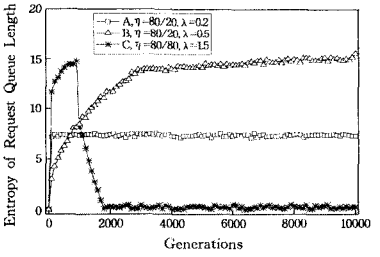


图 6　访问队列长度熵的变化

结束语　本文首先分析网络存储系统中的能耗问题，提出一种改进型的、理想的能耗感知数据分布模型。在此基础上，建立了一个分析网络存储系统能耗与 I/O 性能的元胞自动机模型。网络存储系统的节点能耗状态随着负载的增加，存储节点行为的相关性也逐渐增加，整体性行为表现增强，表现为系统总体的副本数目不断增加并最终趋于稳定，从而系统的总体能耗达到优化。借鉴统计学、动力学以及复杂性理论可以更好地理解网络存储系统的各种复杂性行为，因而本文研究将有助于复杂网络存储系统能耗管理与控制等关键技术的评估和发展。

在未来的工作中，将研究读请求与写请求结合的工作负载下的系统行为。而且，将进一步分析模型演化中存在的各种规律和相应参数的影响，例如分析热点数据的访问不均衡条件下的转移规则、分析多个副本对于各个节点容错能力的影响等。

参 考 文 献

[1] 陈进才，何平，葛雄资. 面向复杂网络存储系统的元胞自动机动力学分析方法[J]. 软件学报，2008，19(10)：2517-2526

[2] Du D. Recent Advancements and Future Challenges of Storage Systems[C]// Proceedings of the IEEE. NJ，USA，2008：1875-1886

[3] Amur K，Cipar J，Gupta V，et al. Robust and flexible power-proportional storage[C]// Proceedings of the 1st ACM Symposium on Cloud Computing. New York，NY，USA，ACM，2010：217-228

[4] 陈茂科，李星. 不完全活动的分组交换格点网络模型的行为[J]. 计算机学报，2005，28(7)：1130-1137

[5] 袁坚，任勇，山秀明. 一种计算机网络的元胞自动机模型及分析[J]. 物理学报，2000，49(3)：398-402

[6] 袁坚，任勇，刘锋，等. 复杂计算机网络中的相变和整体关联行为[J]. 物理学报，2001，30(7)：1221-1225

[7] Ohira T，Sawatari R. Phase transition in a computer network traffic model[J]. Physics Review E，1998，58(1)：193-195

[8] Weddle C，Oldham M，Qian J，et al. PARAID：the Gear-shifting Power-aware RAID[C]// Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07). San Jose，CA，USENIX Association，February 2007：245-260

[9] Mesnier M，Mellon C，Gregory I，et al. Seagate Research，Object-based Storage[J]. IEEE Communications Magazine，2003：84-90

[10] Wolfram S. Cellular automata as models of complexity[J]. Nature,1984,311:419-424

[11] Pinheiro E, Bianchini R, Dubnicki C. Exploiting redundancy to conserve energy in storage systems[J]. ACM SIGMETRICS Performance Evaluation Review, ACM,2006:15-26

[12] Colarelli D, Grunwald D. Massive arrays of idle disks for storage archives[C]// Proceedings of the 2002 ACM/IEEE Conference on Supercomputing. ACM,2002:1-11

[13] Pinheiro E, Bianchini R. Energy conservation techniques for diskarray-based servers[C]// Proceedings of the ACM/IEEE Conference on Supercomputing. ACM,2004:88-95

[14] Narayanan D, Donnelly A, Rowstron A. Write Off-loading: Practical Power Management forEnterprise Storage[C]// the 6th USENIX Conference on File and Storage Technologies (FAST'08). USENIX Association,2008:253-267

[15] Adami C. Introduction to artificial life[M]. New York: Springer-Verlag,1998:94-98

[16] Lee W, Scheuermann P, Vingralek R. File Assignment in Parallel I/O Systems with Minimal Variance of Service Time[J]. IEEE Trans. Computers,2000,49(2):127-140

(上接第 286 页)

对前一代产品 GT200 提升 8 倍，带有 ECC 校验功能，增加了 L1 和 L2 两级缓存，内存带宽提高两倍。Fermi (Tesla C2050)每个 SM 中的寄存器文件数量为 32K[11,12]，对于小规模的 FFT，每个线程拥有的寄存器数量非常充足。因此，在 Fermi 平台，我们对小规模的 FFT 完全展开。此外，Fermi 支持高精度的乘加指令 FMA(fused multiply-add)[11]，FFT 中复数相乘的操作非常频繁，有效使用 FMA 指令优化复数运算能够改善运算精度和运算效率。线程组成 warp 进行访存优化，减少线程之间的同步，同时通过转置操作优化全局内存的对齐访问，对于 block 内部线程之间尽量使用 local memory (对应于 CUDA 中的 Shared memory)来进行 block 内全局通信。在 Fermi 上通过采用如上技术优化 Radix-8，Radix-64 以及 Radix-512，使其性能大大提升，实验结果如图 10 所示。

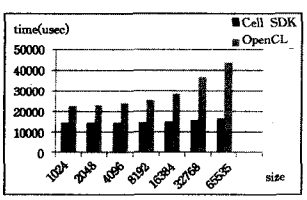


图 7 Cell 平台上的测试结果

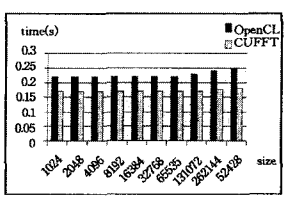


图 8 Tesla C1060 平台上的测试结果

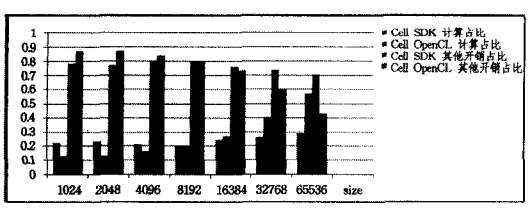


图 9 OpenCL FFT 在 Cell 平台上的计算和开销占比

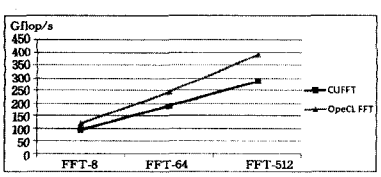


图 10 Fermi 平台上的小规模 FFT 优化结果

结束语 本文测试和分析了面向异构平台基于 OpenCL 的一维 FFT，实验结果表明，在 Cell 平台上当数据规模适中时能达到 SDK 中提供的 FFT 性能的 65%，当数据规模继续增大时，性能有所降低了，仅为 SDK 的 35% 左右。在 NVIDIA 平台上能够到达 CUFFT 性能的 75%，并对小规模的 FFT 在 Fermi 进行了手工调优，性能接近 CUFFT 的 140%。另外，在 Cell 上由于 SPE 具有大容量向量寄存器的特点，下一步工作会利用循环展开和寄存器分块来有效利用寄存器；在 NVIDIA 平台上借鉴 CUDA 程序的优化技术来改善 OpenCL Kernel 中的性能，例如流技术。此外，将针对 Fermi 平台实现和优化更多的小因子 FFT(codelets)，并对一定规模大小的 FFT 采用多种搜索策略和匹配 codelets 算法来得到较优的性能。

总之，OpenCL 为软件开发人员提供了统一的面向异构系统的并行编程环境，尤其在 Cell 上，屏蔽了基于 SDK 编程的很多复杂的硬件细节，提高了并行编程的生产率，因此，OpenCL 在并行计算领域具有良好的应用前景。

结束语 本文在异构平台 Cell 和 Nvidia GPU 上使用 OpenCL 实现和优化了一维 FFT，并测试和分析了该版本 FFT 的性能。此外，针对 Fermi 的存储层次和硬件特点，通过手动调优小因子 FFT，使其性能接近 CUFFT 的 140%。

参 考 文 献

[1] Bracewell R N. The Fourier Transform and Its Applications(3rd ed)[M]. McGraw-Hill,1999

[2] Govindaraju N K, Lloyd B, Dotsenko Y, et al. High Performance Discrete Fourier Transforms on Graphics Processors[C]// SC08. November 2008

[3] Volkov V, Kazian B. Fitting FFT onto the G80 architecture[R]. http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project6_report.pdf, May 2008

[4] Cooley J W, Tukey J W. An algorithm for the machine calculation of complex Fourier series [J]. Mathematics of Computation,1965,19(90):297-301

[5] IBM Corporation. Cell Broadband Engine Programming Handbook[S]. Version 1.12, Apr. 2009

[6] IBM Corporation. Software Development Kit 2.1 Programmer's Guide[S]. Version 2.1, March 2007

[7] IBM Corporation. Cell Broadband Engine Programming Tutorial [S]. Version 2.0, Dec. 2006

[8] Khronos OpenCL Working Group. The OpenCL Specification [S]. Version 1.0, Oct. 2009

[9] AMD Corporation. AMD Core Math Library for Graphic Processors Release Notes for Version 1.0[S]. March 2009

[10] CUDA Programming Guide Version 3.0: NVIDIA Corporation [S]. Feb. 2010

[11] NVIDIA's Next Generation CUDA Compute Architecture: Fermi. NVIDIA Corporation[S]. 2009

[12] Glaskowsky P N. NVIDIA's Fermi: The First Complete GPU Computing Architecture[Z]. September 2009

[13] Frigo M, Johnson S G. The Design and Implementation of FFTW3 [J]. Proceedings of the IEEE,2005,93 (2):216-231

[14] Frigo M. A Fast Fourier Transform Compiler[C]// Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation. Feb. 1999