

EERAIID: Energy Efficient Redundant and Inexpensive Disk Array *

Dong Li, Jun Wang

Department of Computer Science and Engineering
University of Nebraska-Lincoln, Lincoln, NE 68588-0115
{li,wang}@cse.unl.edu

Abstract

Recent research works have been presented on conserving energy for multi-disk systems either at a single disk drive level or at a storage system level and thereby having certain limitations. This paper studies several new redundancy-based, power-aware, I/O request scheduling and cache management policies at the RAID controller level to build energy-efficient RAID systems, by exploiting the redundant information and destage issues of the array for two popular RAID levels, RAID 1 and RAID 5. For RAID 1, we develop a Windowed Round Robin (WRR) request scheduling policy; for RAID 5, we introduce a N-chance Power Aware cache replacement algorithm (NPA) for writes and a Power-Directed, Transformable (PDT) request scheduling policy for reads. Trace-driven simulation proves EERAIID saves much more energy than legacy RAIDs and existing solutions.

1 Introduction

Recently energy has become a critical resource in modern computing thus motivating an upsurge of activity in industry and research. Previous studies show that disk-based I/O subsystems account for a non-trivial portion of energy consumption in both low-end and high-end I/O servers, as evidenced by the fact that 27% of the total energy was consumed by storage devices in a data center [2, 23].

Current low-power storage research work can be categorized into two classes: a single disk mostly used in mobile devices, and multi-disk systems such as RAID in I/O server and data centers. The focus of this paper is on the latter class. In the former class, most solutions share a common idea: extending disk idle duration by shutting down the disk and switching the disk between active and standby modes [7, 8, 11, 12]. In the latter class, many studies [3, 9, 23] have shown that the idle time is too short to apply the simple shut-down policy because the

server I/O workload is much more intensive than that in portable devices. Gurumurthi et al. advocated the use of multi-speed disks and developed power management policy named DRPM to reduce energy consumption for server workloads [9]. They saved energy by creatively fine-tuning multiple-level disk speeds though the multi-speed disk is still not seen at market yet. Carrera et al. studied some hybrid approaches by combining laptop disks and server disks together to conserve energy [3]. Recently Zhu et al. developed several power-aware storage cache management algorithms to indirectly save energy [23]. These solutions work out energy savings at a single disk drive level, a storage system level or OS level. Disk-level solutions do not consider global information about the overall data storage and the role it plays, while high-level solutions do not consider relationships inherent among specific disks in RAID [16]. In both cases, the fact that disks work collaboratively to maintain redundancy and load balance is not accounted for. As a result, current solutions may not perform well and maximize the energy savings.

The authors believe that it is a good opportunity to work at RAID controller level. By conducting I/O request scheduling and cache management with the awareness of both redundancy information and power states of all the disks within the RAID array, I/O requests can be selectively delivered to those high-power state disks so that the idle periods on low-power state disks are arbitrarily stretched. As a result, the aggregate energy consumption is achieved for the entire disk array. The rationale behind this is that, the multi-speed disk has a much higher power consumption rate in high-speed states (i.e., high-power states, active) than that in low-speed states (i.e., low-power states, inactive). If we could make a disk to remain in its low-power state by rescheduling or transforming its I/O requests to other high-power state disks (based on redundant information such as mirroring in RAID 1 and parity in RAID 5), the aggregate energy consumption of the entire disk array can be achieved.

*This work is supported in part by a 2003/2004 Layman grant at University of Nebraska Lincoln.

This paper studies several novel redundancy-based, power-aware, dynamic I/O request scheduling and cache management policies at the RAID controller level, which are used to build Energy-Efficient RAID (EERAID). EERAID would employ either a multi-speed disk or a single-speed (conventional) disk as its basic building block. In this paper, we construct EERAID with the multi-speed disk without specific explanation, for its potential higher energy savings compared to the single-speed disk. For RAID 1, we develop a Windows Round Robin (WRR) request scheduling policy. For RAID 5, we introduce two energy-saving policies, a N-chance Power Aware cache management algorithm (NPA) for writes, and a Power-Directed, Transformable (PDT) request scheduling for reads. Our solutions can be easily extended to other RAID levels, such as RAID 6, RAID 10, RAID 50, etc. By replaying both synthetic and real-life traces in trace-driven EERAID simulators, we found, 1) compared with conventional RAIDs, EERAID 1 and EERAID 5 save up to 74% and 60% energy savings respectively; 2) Compared with DRPM, EERAID 1 and EERAID 5 can conserve up to 19% and 7% extra energy and 3) Both EERAID 1 and EERAID 5 achieve comparable performance of DRPM.

The remainder of this paper is organized as follows. Section 2 introduces the existing disk power models. Section 3 explains the detailed EERAID design. Our experimental methodology and results are presented in Section 4. Section 5 describes the related work and Section 6 discusses the conclusion and future work.

2 Existing Disk Power Model

Modern disks have several modes such as active, idle, standby and etc. Disks in standby mode consume much less energy compared with they do in active and idle modes in both of which disks rotate at full speed. However standby disks need to spin up to a full speed before serving a request. A relatively long time (tens of seconds) is needed to either spin down active/idle disks to standby, or spin up standby disks to active. During the spin-up process, a non-trivial energy is consumed.

The traditional power-efficient way to shut down the active/idle disks by dynamically detecting a threshold of disk idle time is refereed as TPM [9]. As discussed before, TPM can not achieve much benefit in server environments at all where the idle periods are too short to save energy because of the long latency of a complete shut-down and spin-up procedure. They proposed a DRPM scheme, which is based on a multi-rotation-speed disk model where less rotation speed consumes less power. In the multi-

rotation-speed model, the transition time between two different rotation speeds is calculated by their speed distance and thereby *relatively much smaller* compared with that between active/idle and standby modes. Periodically each disk inspects its request queue to decide whether to ramp down its rotation speed or not (namely, the array controller sends a set of operating RPM values to individual disks) based on own performance characteristics.

3 EERAID Design

EERAID aims to develop I/O request scheduling and cache management policies at RAID controller level to conserve energy without introducing too much performance degradation. Since RAID 1 and RAID 5 are two of the most popularly used RAID level organizations, we develop separate solutions used to build EERAID 1 and EERAID 5 in this section.

3.1 EERAID 1

RAID 1, also called mirroring or shadowing, adopts twice as many disks as a non-redundant disk array to maintain 100% redundancy [4]. For RAID 1 controller, there are many alternatives on how to dispatch a read request to disks within the array to achieve high performance [5]. Some commonly used dispatch policies include: 1) (*Primary*), all read requests being sent to a single primary replica; 2) (*Random*), randomly selecting one replica to service a read request; 3) (*Round-Robin*), requests being assigned to replicas in a round-robin fashion; 4) (*Shortest-Seek*), selecting the replica with the shortest seek distance while ties are broken by random selection; 5) (*Shortest-Queue*), selecting the replica with the shortest request queue and ties are broken by random selection. We call all of these policies *general dispatching policy* except *Primary*.

None of the above-mentioned solutions takes the energy consumption into consideration, thus motivating us to develop EERAID 1 by implementing an I/O request scheduling policy called Windows Round-Robin (WRR) dispatch algorithm. The main idea of WRR is to alternatively dispatch every N successive requests to the primary group, then to the mirror group and back and forth. The traditional Round-Robin policy can be taken as a special instance of WRR with N equaling to one. When N requests arrive at RAID 1 controller and are delivered to one group, disks among the other group have more opportunities to continuously ramp down speed and stay longer in low power states. Therefore, the overall energy in the entire disk array is conserved. However, in I/O intensive workload environments, passing N successive requests to one group might stretch the average response time at some situations. To prevent

from compromising too much performance, N should be dynamically adjusted based on performance variance. In EERAID 1, we develop a simple but effective method to intelligently tune the value of N according to the changes of average response time between current request window and previous request window. A request window defines a number (N) of requests to be successively delivered to a disk group.

Algorithm 1: Windows Round-Robin	
(1)	get T by general dispatching policy for N_{max} requests;
(2)	$N = 0$;
(3)	for ($c = 0$; $c < C_{max}$; $c++$)
(4)	{
(5)	if($N < N_{max}$) $N = N + N_{step}$;
(6)	dispatch N requests to the primary group;
(7)	dispatch N requests to the mirror group;
(8)	calculate the average response time T_c of these $2N$ requests;
(9)	$\Delta T = T_c - T$;
(A)	if($\Delta T > p$) goto (1);
(B)	}
(C)	goto (1);

To prevent compromising too much performance, N should be adjusted dynamically to adapt to the online access pattern. We improve our WRR design by increasing the value of N step by step based on the performance variance. First we define a maximum window size as N_{max} . Before running WRR, we calculate the average response time T as a base by using the general dispatching algorithm for N_{max} requests. After then, we select a small number as the initial window size. The RAID controller sends N requests to one disk group and N requests to the other group. These two N request windows are called one window cycle. By comparing T with the average response time of these $2N$ requests, the controller decides whether or not to continue WRR with a larger request window size. If the performance degradation is less than the predefined threshold p (e.g., 5%), the window size is enlarged by the step size N_{step} . Otherwise, WRR is stopped. Even there is no big change on the average response time after many request window cycles, we may still stop WRR and apply general dispatch policy for N_{max} requests to calculate the newest T because the old T may not reflect the current access pattern after such a long time. Let C_{max} be the maximum number of request window cycles. The detail of WRR is illustrated in Algorithm 1. In statements (6) and (7), if a request is a write request, the RAID controller dispatches it to both primary and mirror disk groups.

3.2 EERAID 5

RAID 5 is block-interleaved distributed-parity disk array. It provides a comparable reliability and availability to RAID 1 but require much less storage space. This advantage is at the cost of increasing the number of disk accesses when performing data and parity updates. Generally updating a data stripe involves four physical disk accesses: read old data stripe, read old parity stripe, write new data stripe and write new parity stripe that is generated by exclusive-ORing the first three stripes. The update overhead could seriously degrade system performance, especially in the small-write intensive environment such as on-line transaction processing [14].

Cache is a common solution to resolve this small-write problem [13]. By installing a non-volatile write cache at the RAID controller, disk write operations can be absorbed by the write cache and flushed back to disks as a background activity. Mishra et.al [15] showed that a considerable performance improvement can be achieved by caching both data stripe and parity stripe in a RAID 5 controller. The process of updating cached data and parity information is referred as destaging [20]. Several factors need to be addressed well when designing a good destage algorithm [20]: capturing most rewrites by exploiting the temporal locality in the workload; reducing the number of destages by aggregating physically nearby blocks; conducting accurate idleness prediction to reduce the interference between host reads and the destage accesses and reducing the average latency of destage by ordering requests to the disk.

The destage algorithms that aim to improve performance have been extensively studied, such as least-cost scheduling, high/low mark, linear threshold scheduling [20]. However, few of them took the energy consumption into account. Although recently some cache management policies [21, 23] take the energy consumption as a major concern, none of them works at the RAID system level. Thus none of them see the difference among various disk array organizations. For example, in RAID 5, one disk update operation usually involves more than one disk because of the distributed parity.

To build EERAID 5, a N -chance Power Aware cache replacement algorithm called NPA is developed to preferably destage blocks which host disks are in high-power state (e.g., active, high-speed state) while leaving more cache space to those blocks whose disks are not in high-power state currently. Therefore, the blocks to be destaged to the low-power state disks would stay in the non-volatile cache for a little longer time and thereby obtaining good energy saving on

low-power state disks. NPA works in the following way: firstly, every block has N chances to be replaced. Before a block is destaged by conventional cache replacement algorithm (e.g., LRU), the power states of involved disks are checked. If all engaged disks are in high-power state or the chance of this block is zero, we destage this block. Otherwise, we decrease its chance by one. The detailed procedure of NPA is described in Algorithm 2. NPA can be easily implemented together with most current cache replacement algorithms with the consideration of an additional factor—the power state of destinate disk. In this paper, we develop NPA with LRU.

Algorithm 2: Destage one block in NPA

- (1) check the sorted block list according to the replacement algorithm (e.g LRU);
- (2) let pointer p point to the head of the list;
- (3) if(p arrives at the tail of the list)
- (4) p goes back to the head of the list;
- (5) get the block pointed by p ;
- (6) check the power state of disks involved to destage this block;
- (7) if(all the involved disks are in active states)
- (8) destage this block;
- (9) else if(the chance of this block is not zero)
- (a) {
- (b) decrease its chance by one;
- (c) p points to its next block;
- (d) goto (3);
- (e) }
- (f) else destage this block;

The NPA destage algorithm works only for write requests. For read requests, we develop a Power-Directed, Transformable (PDT) request scheduling policy. At current stage, we only present this idea by an example. Suppose we have a parity group stripe 0, 1, 2 and P , here P is the parity stripe. Stripe 0 and P are cached. When there is a read request for stripe 1, we have two choices to schedule the request at RAID controller: read stripe 1 from its disk, or read stripe 2 from another disk and then generate stripe 1. We make our decision based on the power state of host disks of stripe 1 and 2. We would always try to transform such a read request to a high-power state disk instead of a low-power state disk. The motivation of PDT is similar to that of NPA but the technical concern is that, how much chances it can perform such read request transformations. Through experiments, we found that, PDT can work for more than 20% of all read requests in a RAID 5 with relatively small parity group size (e.g., less than 6 disks), or a RAID 5 with relatively large parity group (e.g., larger than

12 disks) but with large sequential disk request (e.g., more than half parity group size) workloads.

4 Preliminary results

We developed EERAID 1 and EERAID 5 simulators to evaluate our new schemes, and chose DRPM as the baseline system provided by authors of DRPM [9]. We used IBM Ultrastar 36ZX [1, 9] as a building block for EERAID, as shown in Table 1. We built RAID 1 and RAID 5 systems with 12 IBM Ultrastar 36ZX disks. For WRR, we set six to be the maximum window cycle C_{max} . The maximum window size N_{max} and step size N_{step} are 1000 and 200. The performance degradation threshold is 5%. In RAID 5, the stripe size is 8 KB and the data layout is left-symmetric. The RAID controller cache size is 64 MB. The chance N is two.

Table 1: Disk Power Model

DRPM Parameter	Value
Individual Disk Capacity	33.6GB
Max. Disk Rotation Speed	12000RPM
Idle Power at 12000RPM	22.3W
Active Power at 12000RPM	39W
Seek Power at 12000RPM	39W
Standby Power	4.15W
Spinup Power	34.8W
Spinup Time	26 secs.
Spindown Time	15 secs.
Disk-Arm Scheduling	Elevator
Power Model Type	Quadratic
Minimum Disk Rotation Speed	3600 RPM
RPM Step-Size	600RPM

We use both synthetic and real traces to evaluate our design. Based on the reference work [19], all the synthetic workloads consist of 60% read requests and 20% of all requests are sequential. We set 16 KB as the average request size. According to their average interval-arrive time, these synthetic traces are called 50-ms and 200-ms respectively. Two real-world server-side traces are also selected. One is from the Cello96 trace suite (<http://tesla.hpl.hp.com/public/software>) that is collected from the "cello" server over the period from 9 September to 29 November 1996. The other trace is TPC-C20 (<http://tds.cs.byu.edu/tds/index.jsp>), a disk-level trace of running TPC-C database benchmarks with 20 warehouses.

The results of energy saving and performance impact are shown in Table 2 where the second value in parenthesis of the third column is the performance (average response time) impact compared with DRPM. From Table 2, we can tell EERAID 1 and

Table 2: Energy Saving and Performance Impact

RAID 1	DRPM	EERAID 1
50-ms	21.99%	37.85% (-0.38%)
200-ms	55.02%	74.63% (+5.11%)
Cello96	10.18%	10.51% (+1.18%)
Tpcc	15.43%	24.68% (+5.33%)
RAID 5	DRPM	EERAID 5
50-ms	21.45%	29.02% (-0.44%)
200-ms	55.03%	60.09% (+4.25%)
Cello96	3.07%	5.24% (-3.60%)
Tpcc	20.10%	25.79% (+2.87%)

EERAID 5 achieve up to respectively 74% and 60% energy saving compared with legacy RAID organizations. In all the traces, EERAID 1 and EERAID 5 consistently achieve better energy savings than DRPM. This proves WRR and NPA work extremely well. By applying WRR and NPA, EERAID can generate much longer intervals than DRPM since DRPM cannot change the request inter-arrival time on purpose. Compared with DRPM, EERAID 1 conserves up to 19% extra energy saving and EERAID 5 can save 7% extra energy. EERAID 5 conserves less extra energy than EERAID 1 as EERAID 1 maintains a 100% redundancy, which provides a larger read transformation space than that of EERAID 5 in which the redundancy is only 1/G (G denotes the parity group size) and the disk selection is limited by current disk power state.

Both EERAID 1 and EERAID 5 achieve comparable performance with DRPM. WRR and NPA are meant to increase the request burstness and optimize the disk request distribution toward energy efficiency. However, increasing the request burstness might stretch the average response time under heavy workloads, as evidenced by the experimental results of 50-ms and Cello96. As we can see from Table 2, EERAID 1 has little performance degradation in 50-ms while EERAID 5 works similarly in both 50-ms and Cello96 compared with DRPM. But for relatively light workloads, such as 200-ms and Tpcc, both EERAID 1 and EERAID 5 achieve even better performance than DRPM because the switch frequency among different disk speeds is reduced.

5 Related work

Power management research for single disk systems has been extensively conducted in recent years. A lot of work have been done on investigating how to select thresholds to switch the disk between different power modes [7, 8, 11, 12]. Besides these policies

working on disk driver, some researchers studied how to realize I/O power-efficiency at operating system level and application level. Zeng et al. developed an ECOSystem in Linux host OS [22], which unified energy accounting over diverse hardware components and enabled fair allocation of available energy among applications. Their focus is on the entire OS level rather than specific I/O devices. Lu et al. [12] presented a design that allowed the applications to be involved in energy management. Several new system calls were introduced to enable applications to inform OS about future hard disk requests. The Coop-I/O approach [21] is presented to reduce the power consumption of devices encompassing all levels in the computer system. EERAID design is orthogonal to the above approaches. With the combination of the above schemes, EERAID can conserve more energy.

Existing power-efficient scheduling policies for a single disk are not workable for the server system because the duration of disk idleness period is too short for the disk to spin down and spin up [3]. Colarelli et al. [6] used “cache disks” to cache active files/blocks, allowing other disks to spin down. How to conserve energy on network servers has also been studied in references [10, 18]. Pinheiro and Bianchini presented a Popular Data Concentration (PDC) scheme to save energy for network servers by skewing the load toward a few of all the disks [17]. All of the above-mentioned techniques do not work at the RAID system level and thus ignore the details among various disk array organizations. As a result, their approaches either do not work well for making wrong assumptions or could not achieve the best energy-saving. EERAID is able to exploit redundant information and built-in-cache management issues to directly optimize the disk access distribution for all disks with the array.

6 Conclusion and Future work

In this paper, we introduce EERAID which can save significant energy by exploiting RAID redundant information. We present Windows Round-Robin (WRR) dispatch policy for RAID 1. For RAID 5, we introduce a Power-Directed, Transformable (PDT) request scheduling for read requests, and a new N-chance Power Aware cache management algorithm (NPA) for write requests. Preliminary experiment results show EERAID 1 and EERAID 5 can achieve up to 19% and 7% extra energy saving respectively and comparable performance compared with DRPM. In the future, we will firstly implement PDT for EERAID 5 and then develop our EERAID based on single-speed disks.

References

- [1] IBM Hard Disk Drive - Ultrastar 36Z15. <http://www.hitachigst.com/hdd/ultra/ul36z15.htm>.
- [2] Power, heat, and sledgehammer. White paper, maximum Institution Inc., <http://www.max-t.com/downloads/whitepapers/SledgehammerPowerHeat20411.pdf>, 2002.
- [3] E. V. Carrera, E. Pinheiro, and R. Bianchini. Conserving disk energy in network servers. In *Proceedings of the 2003 International Conference on Supercomputing (ICS-03)*, pages 86–97, New York, June 23–26 2003. ACM Press.
- [4] P. M. Chen, E. L. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID : High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, June 1994.
- [5] S. Chen and D. Towsley. A performance evaluation of RAID architectures. Technical Report UM-CS-1992-067, Departement of Computer Science, University of Massachusetts, Amherst, MA 01003 USA, 1992.
- [6] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *SC'2002 Conference CD*, Baltimore, MD, Nov. 2002. IEEE/ACM SIGARCH. CU, Boulder.
- [7] F. Douglass, P. Krishnan, and B. Bershad. Adaptive disk spin-down policies for mobile computers. In *Proceedings of the 2nd Symposium on Mobile and Location-Independent Computing (MLICS'94)*, pages 121–137, Berkeley, CA, USA, Apr. 1995. USENIX Association.
- [8] P. Greenawalt. Modeling power management for hard disks. In *Proceedings of the Symposium on Modeling and Simulation of Computer and Telecommunication Systems (MASCOTS 1994)*, pages 62–66, Jan. 1994.
- [9] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. DRPM: dynamic speed control for power management in server class disks. In D. DeGroot, editor, *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA-03)*, volume 31, 2 of *Computer Architecture News*, pages 169–181, New York, June 9–11 2003. ACM Press.
- [10] T. Heath, B. Diniz, E. V. Carrera, W. M. Jr., and R. Bianchini. Self-configuring heterogeneous server clusters. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP)*, Sept. 2003.
- [11] D. P. Helmbold, D. D. E. Long, and B. Sherrod. A dynamic disk spin-down technique for mobile computing. In *Mobile Computing and Networking*, pages 130–142, 1996.
- [12] Y.-H. Lu and G. D. Micheli. Adaptive hard disk power management on personal computers. In *Proceedings of the IEEE Great Lakes Symposium*, pages 50–53, Mar. 1999.
- [13] J. Menon and J. Cortney. The architecture of a fault-tolerant cached RAID controller. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 76–86, San Diego, California, May 17–19, 1993. ACM SIGARCH and IEEE Computer Society TCCA.
- [14] J. Menon and D. Mattson. Performance of disk arrays in transaction processing environments. In *12th International Conference on Distributed Computing Systems (ICDCS '92)*, pages 302–309, Washington, D.C., USA, June 1992. IEEE Computer Society Press.
- [15] S. K. Mishra and P. Mohapatra. Performance study of RAID-5 disk arrays with data and parity cache. In *Proceedings of the 25th International Conference on Parallel Processing*, volume I, Architecture, pages I:222–229, Boca Raton, FL, Aug. 1996. CRC Press. Iowa State.
- [16] D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, volume 17, pages 109–116, New York. ACM.
- [17] E. Pinheiro and R. Bianchini. Energy conservation techniques for disk array-based servers. In *Proceedings of the 18th International Conference on Supercomputing*, June 2004.
- [18] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power COLP'01*, Sept. 2001.
- [19] C. Ruemmler and J. Wilkes. UNIX disk access patterns. In *Usenix Conference*, pages 405–420, Winter 1993.
- [20] A. Varma and Q. Jacobson. Destage algorithms for disk arrays with non-volatile caches. In H. Jin, T. Cortes, and R. Buyya, editors, *High Performance Mass Storage and Parallel I/O: Technologies and Applications*. IEEE/Wiley Press, New York, 2001.
- [21] A. Weiel, B. Beutel, and F. Bellosa. Cooperative I/O - a novel I/O semantics for energy-aware applications. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI '02)*, Boston, MA, Dec. 2002.
- [22] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat. ECOSystem: managing energy as a first class operating system resource. *ACM SIGPLAN Notices*, 37(10):123–132, Oct. 2002.
- [23] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao. Reducing energy consumption of disk storage using power-aware cache management. In *Tenth International Symposium on High Performance Computer Architecture (HPCA-10)*, Madrid, Spain, Feb. 14–18, 2004.