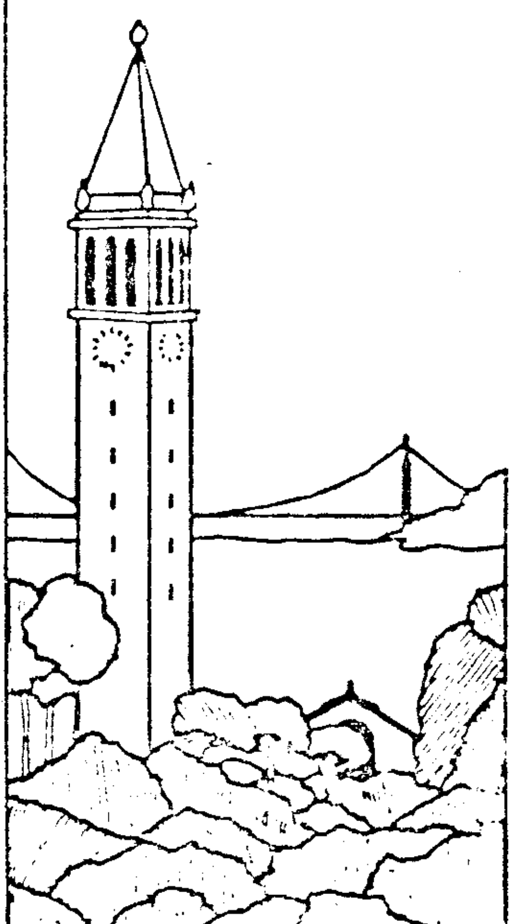
An Evaluation of Redundant Arrays of Disks using an Amdahl 5890

Peter M. Chen



Report No. UCB/CSD 89/506

May 1989

Computer Science Division (EECS)
University of California
Berkeley, California 94720

An Evaluation of Redundant Arrays of Disks using an Amdahl 5890

Peter M. Chen

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California
Berkeley, CA 94720

ABSTRACT

I/O systems are increasingly becoming a major performance limitation to faster computer systems. Recently we presented several disk array architectures designed to increase the data rate and I/O rate of supercomputing applications, transaction processing, and file systems [Patterson 88]. In this paper we present a hardware performance measurement of two of these architectures, mirroring and rotated parity. We see how throughput for these two architectures is affected by response time, request size, and the ratio of reads and writes. We also explore tradeoffs in the unit of interleaving and number of disks. We find that for applications with large accesses, such as many supercomputing applications, a rotated parity disk array far outperforms traditional mirroring architecture. In contrast, for applications with many small accesses, such as transaction processing and traditional file systems, mirroring disk arrays outperform rotated parity disk arrays.

Keywords: disk arrays, performance, I/O, RAID

An Evaluation of Redundant Arrays of Disks using an Amdahl 5890

1. The I/O Crisis

Over the past 10 years, processing speed, memory speed and capacity, and disk capacity have all grown tremendously:

- Single chip processors have increased in speed at the rate of 40%-100% per year [Bell 84, Joy 85]
- Caches have increased in speed 40% to 100% per year
- Main memory has quadrupled in capacity every two or three years [Moore 75, Myers 86]

In contrast, disk access times have undergone only modest performance improvements. For example, seek time has improved only about 7% per year [Harker 81]. If not remedied, this imbalanced system growth will eventually lead to I/O limited systems [Amdahl 67, Kim 87]. Continued improvement in system performance depends in a large part on I/O systems with higher data rate and I/O rate.

One way to increase I/O performance is by using an array of many disks [Kurzweil 88]. By using many disks, both throughput (MB per second) and I/O rate (I/O's per second). can be increased. Throughput can be increased by having many disks cooperate in transferring one block of information; the I/O rate can be increased by having multiple independent disks service multiple independent requests. With multiple disks, however, comes lower reliability. According to the commonly used exponential model for disk failures [Schulze 88], 100 disks have a combined failure rate of 100 times the failure rate of a single disk. If every disk failure caused data loss, a 100 disk array would lost data every few hundred hours. This is intolerable for a supposedly stable storage system. To protect against data loss in the face of a single disk failure, some sort of data redundancy must exist.

This paper analyzes the performance of several disk array redundancy schemes. The performance analysis is based on a set of experiments carried out on Amdahl hardware. In these experiments, we explore several issues:

- What are the basic differences in throughput and response time between the various redundancy schemes?
- For each redundancy scheme, how do different response time requirements affect throughput?

- How does changing the size of an I/O request affect performance of the redundancy schemes?
- How does changing the read/write ratio affect performance of the redundancy schemes?
- What affect does interleaving data in different units have on performance?
- How do the redundancy schemes scale with increasing numbers of disks?

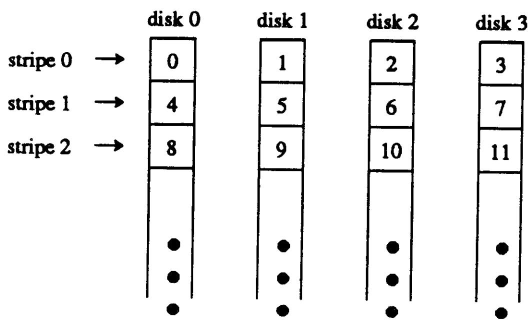
2. Introduction to Redundant Arrays of Disks

In *"A Case for Redundant Arrays of Inexpensive Disks (RAID)"*, henceforth referred to as "The RAID paper" [Patterson 88], Patterson, Gibson, and Katz present five ways to introduce redundancy into an array of disks: RAID Level 1 through RAID Level 5. Using a simple performance model of these five organizations, they conclude that RAID Level 1, *mirrored disks*, and RAID Level 5, *rotated parity*, have the best performance potential. This paper focuses on these two RAID Levels, plus the additional RAID Level 0. RAID Level 0 is a *non-redundant* array of disks, and is added mainly to provide a basis of comparison between RAID Levels 1 and 5. Figure 1 shows the data layout in the three redundancy schemes. The rest of this section summarizes the RAID Levels--see [Patterson 88] for more details.

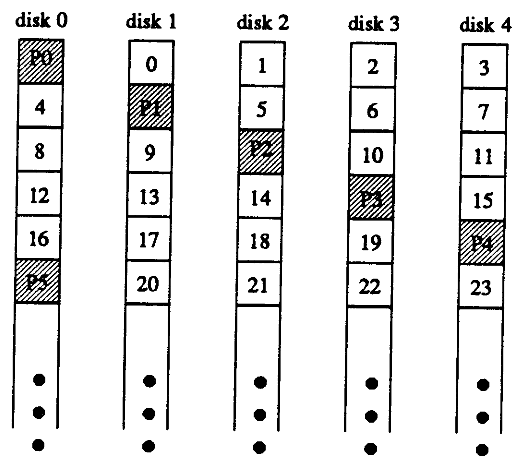
In all organizations, data are interleaved across the disks [Kim 86, Salem 86]. We define a *stripe* of data to be one unit of interleaving from each disk. For example, the first stripe of data in Figure 1 consists of logical blocks 0, 1, 2, and 3. The *storage efficiency*, a measure of the capacity cost of redundancy, is defined to be the *effective (user) data capacity divided by the total disk capacity*. For RAID Level 0, the effective data capacity equals the total disk capacity, so the storage efficiency is 100%.

RAID Level 1, mirrored disks, is a traditional way to incorporate redundancy in an array of disks [Bitton 88]. In RAID Level 1, each datum is kept on two distinct disks: a data disk and a shadow disk. Thus, for RAID Level 1, the effective storage capacity is half the total disk capacity and the storage efficiency is 50%. Reads can be serviced by either the data disk or the shadow disk, but, to maintain consistency, writes must be serviced by both data and shadow disk.

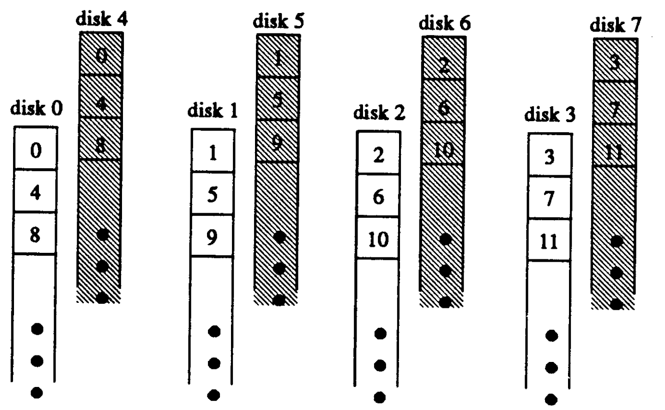
RAID Level 5, rotated parity, incorporates redundancy by maintaining parity across all disks. For example, P0 in Figure 1b is the parity of logical blocks 0, 1, 2, and 3. Parity will have to be updated whenever data is written. If all parity information was kept on one disk, this disk would see many more requests than any data disk. To avoid a bottleneck in accessing the parity information, it is spread over all disks.



(a) RAID Level 0--non-redundant array



(b) RAID Level 5--rotated parity. The shaded areas are parity.



(c) RAID Level 1--mirroring. Shadow disks are shaded.

Figure 1: Three RAID architectures. Data (logical blocks) are interleaved across multiple disks with various redundancies added. In RAID Level 0 (Figure 1a), no redundancy exists. Each stripe of data consists of a logical block from each disk. In RAID Level 1 (Figure 1c), each data disk (disks 0-3) has a shadow disk (disks 4-7). In RAID Level 5 (Figure 1b), parity for each stripe is kept in a parity block. Which physical disk the parity block is kept on is different for different stripes.

There are many ways to spread this parity information across disks, but this is not within the scope of this paper. Instead, we have chosen one mapping of parity information onto disks. As shown in Figure 1b, parity for stripe 0 is kept on disk 0; parity for stripe 1 is kept on disk 1, and so on. If there are N disks in the array, the storage efficiency is $\frac{N-1}{N}$.

3. A Simple Performance Model

In this section we present the simple performance model used by the RAID paper to compare RAID Levels. First, some terminology is needed. The user request to read or write data is called a *logical* request. A *physical* request refers to a logical request after it has been mapped onto the disk array. Often, due to redundancy information, the physical request will involve more disk blocks than the logical request. A *disk access* refers to one contiguous read or write of one disk. Physical requests result in one or more disk accesses. A logical request that involves all the data in a stripe is called a *full stripe* request. A logical request that involves only part of the data in a stripe is a *partial stripe* request. A special type of partial stripe request is an *individual* request, which is a request to exactly one disk's part of a stripe.

The model in the RAID paper is concerned with the maximum possible throughput of a disk system. The model drives the disk system with four types of logical requests: full stripe reads, full stripe writes, individual reads, and individual writes. To estimate maximum possible throughput, we consider the *efficiency* of a RAID: the number of disk accesses of a logical request divided by the number of disk accesses in its corresponding physical request.

Because RAID Level 0 has no redundant information, the number of disk accesses in a physical request is always the same as in its logical request. Thus RAID Level 0 has an efficiency of 100%, that is, 100% of the disk accesses involve useful data. We normalize throughput of a RAID by defining *relative throughput* of a RAID system running a particular workload as the throughput of that RAID system relative to the throughput of a non-redundant array (RAID Level 0) running the same workload (matching workloads will be described later). The simple model estimates relative throughput by the fraction of disk accesses involving useful data. For example, if the physical request involves twice as many disks accesses as its corresponding logical request, i.e. the logical to physical mapping doubled the number of disks accesses involved, then 50% of the disk accesses would involve useful data and the simple model would estimate the relative throughput to be 50%.

For all the RAID Levels that we are concerned with, assuming no failed disks, data can be read without accessing any redundancy information (these experiments do not measure performance when one or more disks are not operational). Because of this, the mapping from logical read requests to physical requests adds no extra disks, and the simple model predicts a relative throughput for RAID Levels 0, 1, and

5 reads of 100%. As mentioned above, RAID Level 0 also has, by definition, a relative throughput of 100% for writes. However, for RAID Levels 1 and 5, physical write requests involve more disk accesses than their logical write requests, and relative throughput becomes less than 100%.

To write data in RAID Level 1, both the data disk and the shadow disk must be written. Thus, a RAID Level 1 physical write request has twice the number of disk accesses as the corresponding logical request. The simple model estimates relative throughput at 50% for any size RAID Level 1 write.

For RAID Level 5, both the data disk(s) and the parity information need to be updated. To compute the new parity, some reads may need to be issued. Some of these reads snapshot the image on disk before those blocks are overwritten. We call these *pre-reads*. How much information needs to be read depends on the size of the logical write request. For full stripe writes, no reads are needed, since the new data completely determines the new parity of the stripe. Thus, with an N disk array, a full stripe logical write request involves N-1 disk accesses, and the physical request involves N disk accesses. This leads us to estimate relative throughput as $\frac{N-1}{N}$. For partial stripe writes, parity may be computed either by 1) pre-reading the current (before writing) data on the data disk(s) and current parity of the stripe or 2) reading the current data in the rest of the stripe. For example, in Figure 1b, to write logical blocks 0 and 1, we can either 1) pre-read logical blocks 0 and 1 and parity block P0 or 2) read logical blocks 2 and 3. With a partial stripe write of D (less than N-1) data disks, the first method of computing parity involves D+1 disk pre-reads, and the second method involves $N-(D+1)$ disk reads. For an individual stripe request ($D=1$) the first method is better for $N \geq 4$. With this first method, an individual request, involving one disk access, generates a physical request involving four disk accesses (two to read the current data and current parity and two to write the new data and new parity). This leads us to estimate relative throughput as 25% for individual stripe writes in RAID Level 5.

The estimates for full and individual requests for both RAID Levels 1 and 5 are summarized in Figure 2.

4. Goals and Refinements

Our overall goal is to understand more fully how RAID Levels 1 and 5 perform. This includes exploring aspects of implementation, workload characterization, and performance evaluation. In

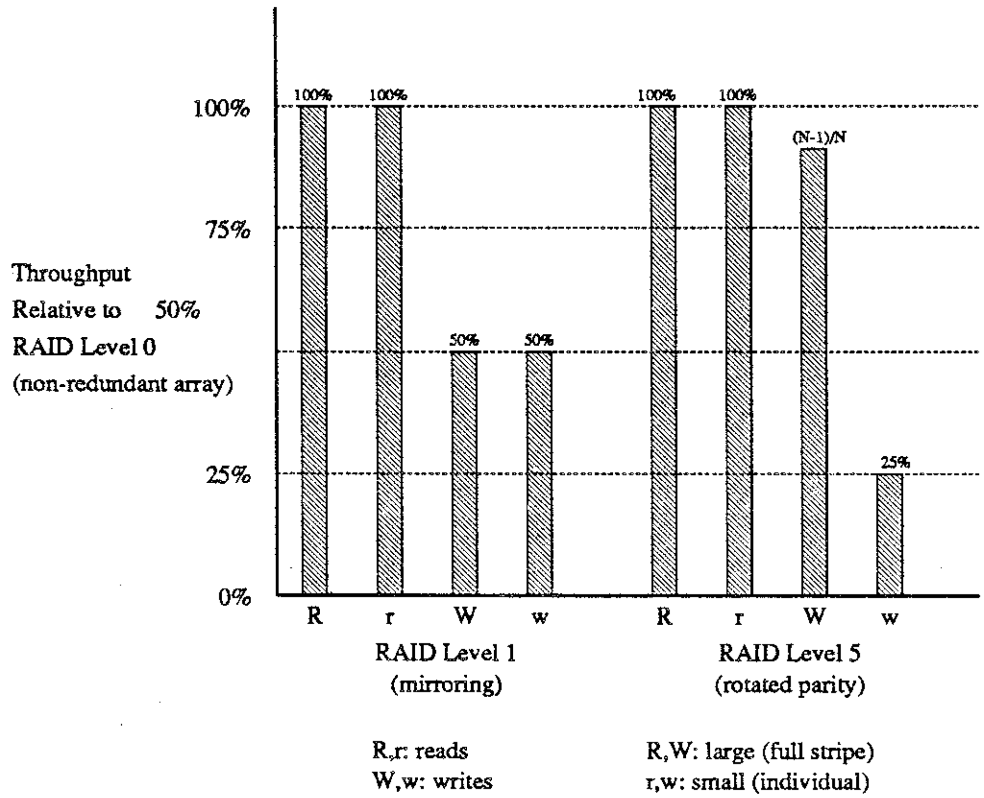


Figure 2: Relative Throughput According to a Simple Model. Shown is the estimated throughput of RAID Level 1 (mirrored RAID) and RAID Level 5 (rotated parity RAID) as a percentage of the estimated throughput of RAID Level 0 (non-redundant RAID) [Patterson 88]. RAID Level 5 large write performance is calculated assuming 11 total disks (*N=11*).

particular, the performance estimates above dealt with maximum possible throughput, expressed in terms of relative throughput. A specific goal of the experiments described here is to *measure* the performance of RAID Levels 1 and 5 and to compare this measured performance against the simple performance estimates in the RAID paper. The performance characterization in these experiments differs from the RAID paper in the following areas:

- *Real hardware*: The analysis done in the RAID paper was a purely theoretical analysis. It assumed a constant time for disk accesses and ignored processing overhead. Because these experiments were carried out on an actual machine with disks, they have no need to make any of these simplifying assumptions.

- *Response time*: The only performance metric in the RAID paper was maximum possible throughput. These experiments will measure and control both throughput and response time.
- *Synthetic workload*: Because the analysis in the RAID paper was theoretical, it used extremely simple, therefore unrealistic, workloads. These experiments refine the workloads in three ways:

    (1) The RAID paper workloads had a constant logical request size of either 100% full stripe requests or 100% individual requests. These experiments deal with a distribution of request sizes, including partial stripe accesses and accesses larger than a full stripe.

    (2) The RAID paper workloads were either 100% reads or 100% writes. These experiments explore a range of read/write ratios.

    (3) The RAID paper used an infinite workload, i.e. the disks were fully utilized. These experiments introduce contention, resulting in more realistic (suboptimal) disk utilization.

Note that these experiments are not running application programs (benchmarks), but rather an artificially generated distribution of I/O requests (synthetic workload). We choose to use synthetic workloads because they are easier to parameterize than benchmarks, making it possible to explore a range of different user workloads. Also, running application programs require an underlying file system, which we do not have.

5. Comparing RAID Levels

When comparing RAID Levels, we are interested in performance (throughput and response time) and cost. Comparing RAID Levels in these two areas is no easy task. Because the storage efficiency differs between RAID Levels 0 (100%), 1 (50%), and 5 ($\frac{N-1}{N}$), RAID systems with the same user data capacity need different numbers of disks. Alternatively, with a fixed number of disks, different RAID Levels will have different user data capacities. There are at least two ways to address this issue.

5.1. Constant Number of Total Disks

The first option is to keep the total number of disks constant between RAID Levels Comparing costs with this option is trivial. Since all RAID Levels use identical hardware, cost is equal. However, comparing the performance of such RAID systems is tricky. To have a valid basis for comparison, equal workloads must be presented to all systems. Unfortunately, it is unclear how to define an equal workload between systems with different data capacities. For example, consider two disk systems, A and B, both with the same number of total disks. Suppose that system A has a storage efficiency of 50% and system B has a storage efficiency of 100%. Thus, system B has twice the user data capacity as system A. If A and B receive the same workload, then the data in B is accessed half as frequently as the data in A. To compensate, we may wish to present B with double the workload that A receives. Unfortunately, it is unclear what constitutes double a workload: double the request rate? Double the request size? Double the unit of interleaving? The second option in comparing RAID Levels takes a different approach to resolve this difficulty.

5.2. Constant Number of Data Disks

The second option, which is the one used in these experiments, maintains equal data capacities between RAID Levels. Presenting identical workloads to each system makes comparing performance far simpler. However, with D disks of user data, RAID Level 0 needs D total disks, RAID Level 1 needs $2D$ total disks and RAID Level 5 needs $D+1$ total disks. Thus, costs and raw disk bandwidth of the different RAID Levels are no longer equal. We must therefore factor in costs when presenting performance.

One method to factor in costs is to simply present the raw performance and cost separately For example, in one of the experiments, a RAID Level 1 used 20 disks and yielded a throughput of 20 MB/s. The corresponding RAID Level 5 system used 11 disks and yielded a throughput of 10 MB/s. In general, RAID Level 1 needs nearly twice as many disks as RAID Level 5 and has much higher cost. Then, having more disks, RAID Level 1 generally yields higher performance than RAID Level 5. It then becomes the reader's responsibility to synthesize this performance and cost data.

A second method to combine performance and cost is to divide the performance by the number of disks. As this only makes sense for throughput, response time will be addressed separately. RAID Level 1 has twice as many disks as RAID Level 0, and so we divide RAID Level 1 throughput by two to normalize relative to RAID Level 0. By dividing the throughput by the number of disks, we are tacitly assuming that a RAID Level 0 with $2D$ disks should perform twice as well as a RAID Level 0 with D disks (see section 10.3). This assumption can be *false* if performance is not *disk limited*. In general, we may be unfairly penalizing RAID Level 1 because we are not providing RAID Level 1 with twice the total resources (processor, memory, etc.) of RAID Level 0. We are only doubling the disk resources. In particular, if CPU power is the limiting factor to performance, then doubling the disk resources will not double throughput. CPU power tends to limit performance as more disks are used. Therefore, we limit the number of disks used to ensure that CPU power does not greatly impact performance and RAID Levels with more disks are not unfairly penalized. We will limit almost all experiments in this paper to 20 disks or less. To check that this method of presenting data is fair, we verify that throughput per disk remains constant as we scale up the number of disks used (see section 10.3). This second method of combining throughput and cost is the one used in this paper.

Although throughput scales with the number of disks used, response time does not. Unless otherwise mentioned, response time in this paper is defined as the time in which 90% of the requests in the run were serviced, similar to [Anon 85]. For example, a response time of 1 second would mean that 90% of all requests in that run returned to the user within 1 second. To maintain a valid comparison between RAID Levels, we vary the rate of requests and force different RAID Levels to have the same response times. With this equal response time, we then compare the throughput of the different RAID Levels.

In summary, we compare different RAID Levels by:

(1) maintaining equal user data capacity to simplify the equalization of workloads

(2) forcing comparable response time for all RAID Levels

(3) measuring throughput and dividing by the number of disks involved to get throughput per disk as the main performance metric.

5.3. Two Entire Systems

As a final note, another approach to comparing RAID Levels is possible. We show this option in Figure 3. This approach is easiest to explain for comparing RAID Level 0 and 1, but it also generalizes to RAID Level 5.

Consider a system with *D* disks and one CPU. Conceptually, we will form a RAID Level 1 with two such systems, systems A and B. Each system in the RAID Level 1 will contain half the data disks along with their corresponding shadow disks. Requests to the RAID Level 1 as a whole will be serviced in part by each of the two systems. Requests that span more than one disk will be broken up and serviced in part

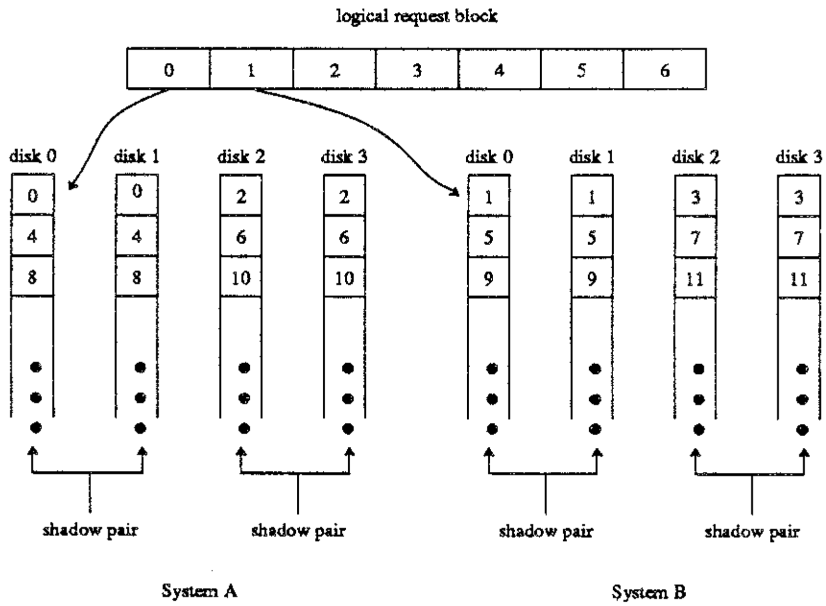


Figure 3: Alternate Way to Compare RAID Levels. This figure shows a way to map RAID Level 1 onto two distinct systems A and B. Even numbered logical blocks are stored in System A; odd numbered logical blocks are stored in System B. Each system will receive exactly half the total workload. Thus, to estimate total throughput, we need only measure the throughput of one of the systems. This has the advantage of needing only half the disk and CPU resources. Also, the system that we measure will have the same CPU resources per disk as a RAID Level 0.

by system A and in part by system B. Similarly, half the requests that need only one disk will be serviced by system A and half will be serviced by system B. The key to this method is that each system will see an identical load. Because of this, we need only measure the throughput of one of these systems and multiply by two to calculate the throughput of the entire RAID Level 1. Note that the conceptual system has double the entire hardware configuration of a corresponding RAID Level 0 system, not just double the number of disks. Because of this, the cost of the resulting RAID Level 1 is exactly double the cost of the corresponding RAID Level 0 and dividing the resulting calculated RAID Level 1 throughput by two will always yield a fair throughput per cost figure.

This method, though superior in comparing throughput per cost, does not accurately model response time. We could assume response time is the same on both halves, as they do the same work. However, this ignores unsynchronized disks. Because of this difficulty in measuring response time, we choose to use the previous method (constant number of data disks).

6. Experiment Implementation

Experiments were run on an Amdahl mainframe under UTS, which is a version of System V Unix. See Table 1 and Figure 4 for hardware statistics and channel architecture. Note that we have only one disk per string. This prevents channel conflicts and approximates a system which uses buffers to avoid data transfer conflicts.

By using synthetic workloads instead of application software, we eliminate the need for a file system structure. Instead, we simply read and write bytes on the disks. This enables us to simulate a large range of I/O access patterns without dealing with the logistics of many benchmarks. In our experiment, the reads and writes are done by user processes accessing raw devices [UTS 88].

6.1. Process Structure

The simplest structure to produce a synthetic workload would be a single process issuing all I/O's. However, because UTS does not support asynchronous I/O, it is impossible to have more than one outstanding I/O per process. Thus, at the start of each experiment, one master process (the parent) creates one child process for each disk. This child process will be used as a user-level device driver and will drive one disk. All communication is done via IPC messages between the parent process and an individual child

Processor Resources: Amdahl 5890-300e	
processors	2
cycle time	15 ns
memory size	256 MB
data cache	64 KB
instruction cache	32 KB
channels	64

Disk Resources: Amdahl 6380	
cylinders/disk	885
tracks/cylinder	15
sectors/track	10
bytes/sector (fixed format)	4 KB
average seek	15 ms
average rotational latency	8.3 ms
maximum transfer rate	2.4 MB/s
disk caching	off

Table 1: Hardware Statistics of Processor and Disk Resources [Amdahl 6380, Amdahl 5890]. We assume rotational latency is distributed uniformly between 0 and 16.7 ms. We also assume seeks are distributed uniformly between 8 ms and 27 ms, an approximation of the seek time data in [Thisquen 88].

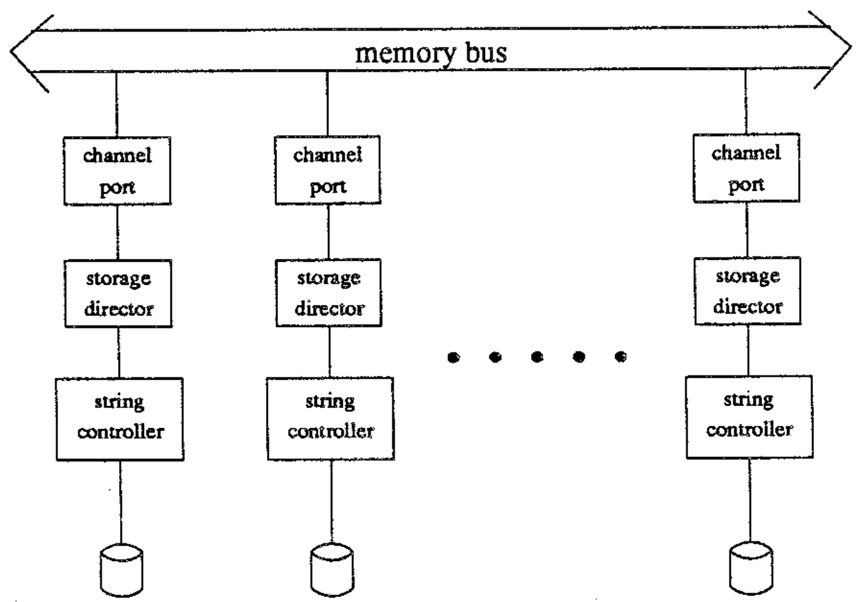


Figure 4: Disk - Memory Bus Architecture. This figure shows how the disks are connected to the memory bus in most of our experiments [Buzen 86]. Note that we use only one disk per channel path (path to memory). This avoids channel conflicts and RPS misses.

process--there is no communication between child processes. The child process has no concept of RAID Levels. Rather, the parent is responsible for generating the logical request, mapping the logical request into a number of physical requests, and passing the physical requests to the children. The children see simply a stream of disk accesses of the form: read/write, location on disk, size of access.

Each child process has a queue of requests, and waits until there is a request in its queue to carry out that request on its assigned disk (using UNIX read or write). When the request returns, the child will inform the parent of its return and process the next request in its queue. Note that no actual data passes between the parent and the child. All data is read into a garbage buffer and thrown away by the child. Data to be written also comes from a garbage buffer.

6.2. RAID Level 0

For a non-redundant array, individual logical reads and writes each result in a single disk access. Because there is no read buffering on the disks we are using, physical reads and physical writes have the

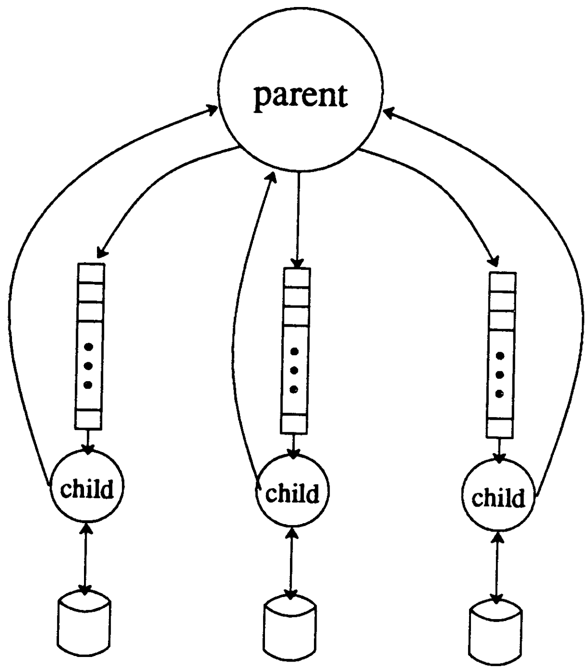


Figure 5: Parent - Child Communication Structure. This figure shows how the parent process communicates with the child processes to issue and return disk accesses. The parent (master) process asynchronously issues disk commands to a queue for each child process, which act as user-level device drivers. In turn, the child processes synchronously issue commands to a disk.

same disk service time. Consequently, in this experiment, we assume performance for RAID Level 0 is independent of the percentage of reads and writes.

6.3. RAID Level 1

To execute a mirrored read of a data block, the system must decide if the data disk or the shadow disk will carry out the access. In our experiment, we first look for a disk which is idle. If both disks are idle or both are busy, we choose the disk which will yield the shorter seek. To make this choice, we save the location of the previous request to that physical disk.

A possible optimization which do not make is to have a read serviced in part by a data disk and in part by a shadow disk. This implementation would increase throughput if the system was fairly idle, since it would make better use of both copies of each datum. However, it would decrease throughput for a loaded system, since each disk would transferring less data per seek.

6.4. RAID Level 5

A RAID Level 5 write must update both the data blocks and the parity associated with those data blocks. Because a full stripe write will write an entire stripe, it involves all the data needed to compute parity, and no additional information needs to be read.

For partial stripe writes, two interesting implementation questions arise. First, which disks should we read to compute parity? One choice is to read the current data and current parity, compare the current data and new data, and change the parity correspondingly. When writing D disks of data out of N total disks, this choice results in $D+1$ disk reads to compute parity. The second choice is to read the data in the rest of the stripe and simply recompute parity of the stripe. This results in $N\text{-}D\text{-}1$ disk reads to compute parity. In our experiment, we choose the option that requires the fewest disk accesses. Partial stripes requests can range in size from 1 disk to $N\text{-}2$ disks. For an individual request, we read the current data and the current parity. This leads to a relative throughput of 25%. For a partial stripe write of $N\text{-}2$ disks, we read the remaining data disk and write the N-2 original data disks plus parity. This leads to a relative throughput of $\frac{N-2}{N}$ (N disks need to be accessed in RAID Level 5 relative to $N\text{-}2$ disks in RAID Level 0).

Second, if we choose to read the current data and current parity, how do we schedule the new data and new parity writes? The new data can be written immediately after the current data has been read. This new data write will see a zero seek plus a full rotation. We assume the kernel can do the exclusive-ors necessary to compute the new parity block in the time the disk rotates once. Thus, we write the new parity block out one rotation after the current parity block has been read (exactly the same as the data block). This is a simplifying assumption, as the current data may not have been read yet. We believe this assumption is valid, as the device driver could simply delay the parity block read and write until the data block is about to be read. Because a request is not considered complete until all accesses related to that request have finished, such a minor scheduling delay would only marginally affect response time and should not affect throughput at all.

6.5. Response Time Control and Stabilization

To achieve a certain target response time, we control the number of outstanding logical requests in the system (*queue depth*). We periodically check the number of requests that were satisfied within the target response time, and adjust the allowed number of outstanding logical requests accordingly. To avoid including the start up time in the performance analysis, we discard statistics gathered before the queue depth has stabilized. Queue depth stabilization in this context means being checkpointed at the same value more than twice. Once the queue depth has stabilized we begin collecting data. The throughput for the run is collected every 20 seconds. The run has stabilized when two conditions are met:

(1) The previous two throughput reports are within 1% of the current throughput report.

(2) 90%±1% of all requests have been fulfilled within the target response time.

Once the run has stabilized, we cease sending new requests and await the completion of any outstanding requests. While we await the completion of all requests, throughput will drop, though not enough to significantly affect the overall performance.

6.6. Synthetic Workload Implementation

The parameters of the synthetic workload are response time target, read/write ratio, request size distribution, and data distribution. When the parent process generates a request, it stochastically chooses read or write, request size, and starting location of the data. Each choice is made independently of past choices. Note that these parameters determine the logical request stream and are independent of the RAID organization (the logical to physical mapping).

The request size is generated in a number of different ways, depending on the workload. One method is to force all accesses for a run to be a fixed size. This is the assumption used in the simple model, for example, 100% full stripe requests. A second method is to choose a distribution of request sizes. We choose two distributions in particular: exponentials with small means and normals with large means and standard deviations. Once we choose the request size, we choose the placement of this data according to the data distribution.

Because there is no file system structure, we choose the data distribution by choosing the starting location of the *logical* request. In our workload, we break the starting location into two orthogonal components: starting disk and stripe number on that disk. The starting stripe number on the disk is always distributed uniformly over all stripes on that disk.

We used several methods for choosing the starting disk. We call the first method *uniform alignment*. This is intended to be the data distribution which yields optimal performance. Uniform alignment tries to 1) minimize the number of partial stripes, and 2) spread the I/O load evenly across disks. It minimizes the number of partial stripes by aligning data on full stripe boundaries where possible. For request sizes smaller than a full stripe, it chooses the starting disk according to a uniform distribution.

The second method of choosing the starting disk is derived from a normal distribution (Figure 6). In our tests, the standard deviation (in disks) of our normal distribution equals the number of data disks. We refer to this data distribution as the *skewed* distribution.

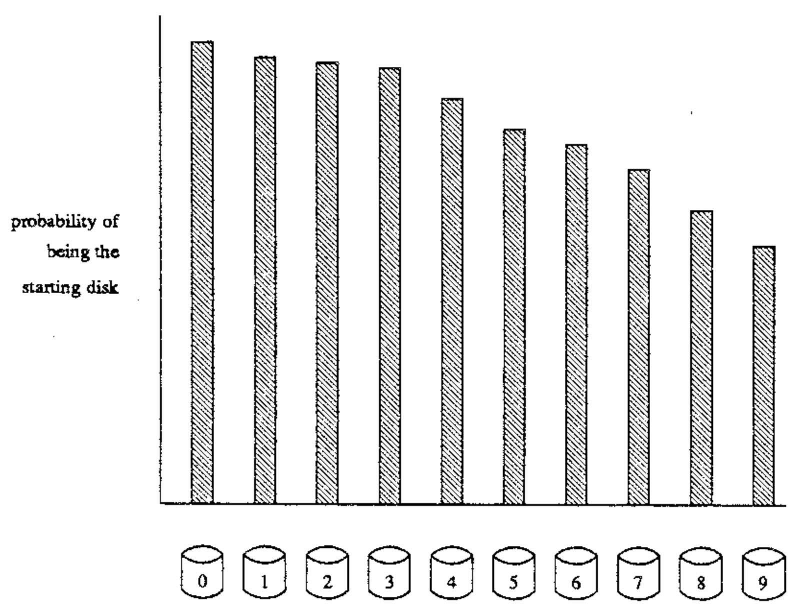


Figure 6: Using the Skewed Data Distribution to Select the Starting Disk. The distribution of data accesses is determined by both starting disk and stripe number on that disk. The choice of starting stripe is always distributed uniformly over all stripes. Starting disk is chosen according to various distributions. The distribution shown here, the skewed data distribution, is the truncated right half of a normal distribution with a standard deviation of 10 disks.

7. Result Presentation

The simple model analyzed performance under four types of workloads: large reads, large writes, small reads, and small writes. While continuing to direct our experiments toward understanding these four types of workloads, we seek to make them more realistic as discussed in Section 4. Rather than jump from the analysis presented in the RAID paper directly to our most realistic workload, we make the transition in a number of smaller steps. These steps are summarized in Table 2.

We start by running the same workload as the RAID paper: request sizes are full stripe or individual; requests are all reads or all writes. In Section 8.1, we analyze an idle system to break down the time for a basic I/O. Next, in Section 8.2, we attempt to duplicate the assumptions made in the RAID paper of unlimited response time by analyzing a saturated system. In 8.3, we make the experiment more realistic by controlling and equalizing the response times.

When we have finished analyzing the RAID paper workload, we begin to use more varied workloads. We again make this transition by changing one aspect of the workload at a time. In 9.1, we remove the assumption of constant request size by using a distribution of request sizes. In 9.2, we skew the data distribution. As our last step in making the experiment more realistic, we allow workloads with both reads and writes.

Section	Description	Response Time	Request Size	Data Distribution	Read Write Mixture
8.1	Breaking down a basic request	minimum	fixed	uniform aligned	unmixed
8.2	Unlimited response time	unlimited	fixed	uniform aligned	unmixed
8.3	Set target response time	target	fixed	uniform aligned	unmixed
9.1	Distribute request sizes	target	distributed	uniform aligned	unmixed
9.2	Skewed data distribution	target	distributed	skewed	unmixed
9.3	Mix reads and writes	target	distributed	skewed	mixed

Table 2: Stages of Results. A guide to the results in Sections 8 and 9. The workload becomes more realistic with each succeeding stage. In Sections 8.1-8.3, we concentrate on the response time target. In Sections 9.1-9.3, we change the request size, data distribution, and read/write ratio.

Lastly, we explore several additional issues, such as further varying the request sizes, comparing sector and track interleaving, and scaling the number of data disks. As a final caveat we look briefly at the effects of connecting multiple disks per channel.

Unless otherwise noted, experiments are run with 10 data disks and track striping. Thus, a full stripe is 10 tracks (400 KB) and an individual request is 1 track (40 KB).

8. The RAID Paper Workload

Request sizes in the RAID paper workload are large (a full stripe) or small (individual). Runs are either 100% reads or 100% writes. With the RAID paper assumption of infinite workload, disks are always 100% utilized, and data distribution has no effect. To approximate this, we use a very high workload and the uniform aligned data distribution.

8.1. Analyzing an Idle System

To understand the supporting hardware, we first break down the time of a basic I/O. This is done by analyzing the average response time of a single request in an idle system. Average response time is *not* the 90% response time used in the majority of this paper, but rather the *arithmetic average* of all response times. We trace the lifetime of an average request by measuring the time spent in various stages of servicing the request (Figure 7). We also measure the total average response time of the requests and check that this is equal to the sum of the times spent in each stage. In all cases, the average response time is within 1 ms of the sum of the time spent in each stage.

Average response time breaks down as follows:

- *request overhead*: CPU time spent in sending messages between parent process and child processes.
- *IO CPU time*: CPU time for children to issue and receive I/O's, including the channel processing time. IO CPU time ranges from 1.5 ms to 1.8 ms.
- *disconnect time*: time spent in seek and rotational latency. An average seek is 15 ms and the average rotational latency is 8.3 ms.
- *synchronization*: additional time due to multiple independent disks doing random seeks and rotations. This is not measured, but rather calculated (more in the next section) based on statistics given in Section 6.

- *connect time*: time spent in transferring data [IBM 87]. For full track data transfers, connect time is 16.2 ms.

Note that request overhead is measured *per logical request.* IO CPU time, disconnect time, pending time, and connect time are all measured *per disk access.*

8.1.1. Synchronizing Multiple Disks

There are two types of synchronization between multiple disks. In most cases, such as RAID Level 0 full stripe reads or writes and RAID Level 1 individual writes, only rotations need to be synchronized. Seek distance among disks that cooperate in a request are usually the same--the sole exception being RAID Level 5 small writes. Because workloads are homogeneous, disks that cooperate in any request cooperate for all requests. For example, for RAID Level 1 individual writes, each data or shadow disk pair always seek to the same track. Thus, for RAID Level 0 full stripe reads or writes, RAID Level 1 full stripe reads or writes and individual writes, and RAID Level 5 full stripe reads or writes, multiple rotations lengthen the total request time in proportion to the number of disks involved. Synchronizing N multiple rotations is equivalent to taking the maximum of N uniform random variables distributed between 0 and 16.7 ms. The expected value of such a distribution is $\frac{N}{N+1} \times 16.7$ ms. The difference between this expected value and the average rotational latency of one disk (8.3 ms) is the penalty for synchronizing multiple rotations.

The second type of synchronization, synchronizing both seeks and rotations, is relevant only for RAID Level 5 individual writes, and is discussed in Section 8.1.4.

8.1.2. Request Overhead

Request overhead changes drastically with the number of disks involved in a request. For example, for RAID Level 0, request overhead jumps from .7 ms for individual reads/writes to 10.9 ms for full stripe reads/writes. This drastic increase is a side effect of running on an idle system. Under normal loads, most of this overhead is overlapped with queuing time and does not noticeably affect performance. In an idle system, when the first of multiple messages is passed from the parent process to a child process, the message is immediately received and acted upon by the child (since the system is idle). As that child processes the I/O request it has just received, it contends for the CPU with the parent process, which is trying to send

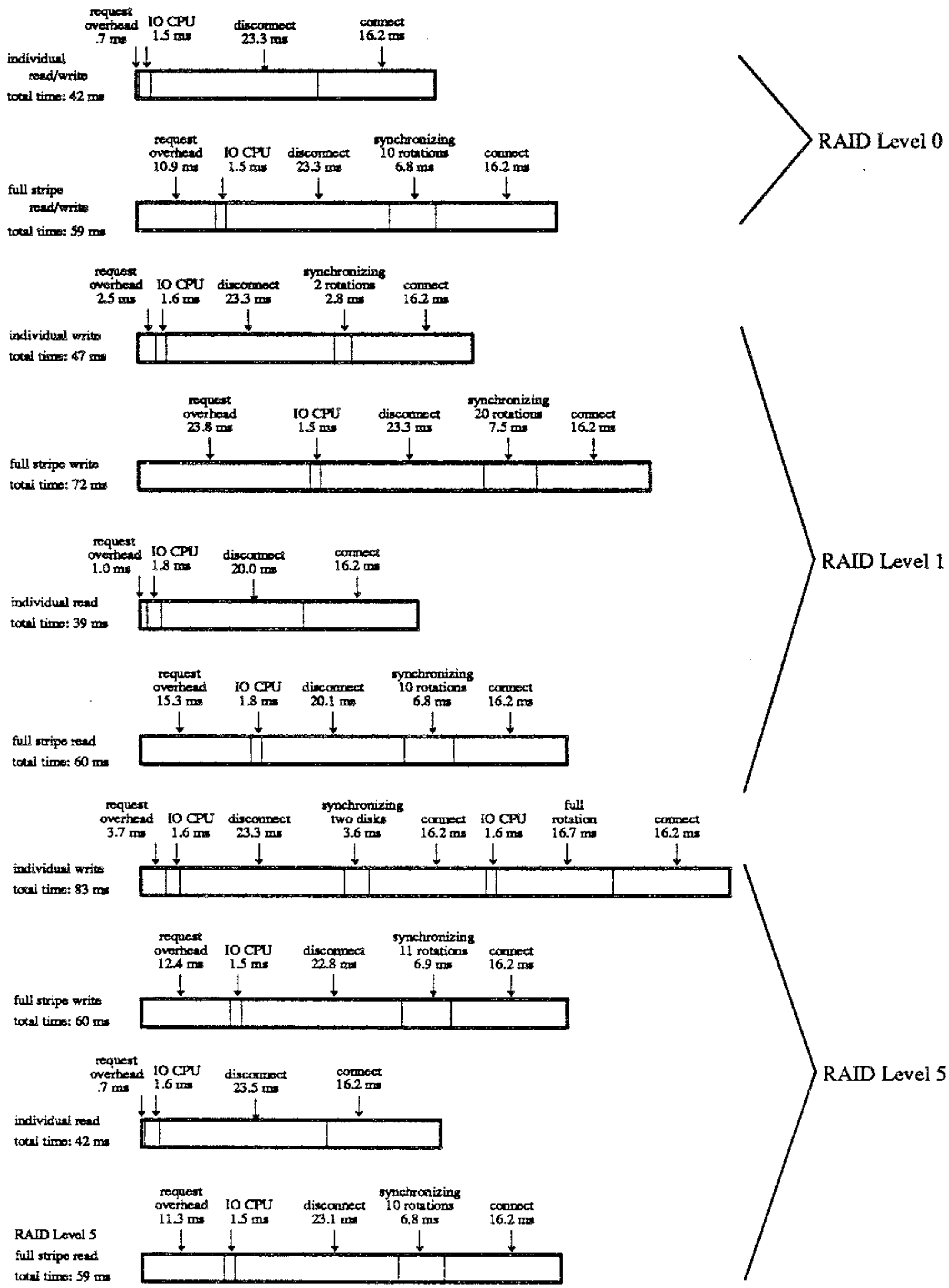


Figure 7: Lifetime of Requests. The lifetime of each type of request is traced by measuring the time spent in various stages of servicing the request: request overhead, IO CPU time, disconnect, synchronization, and connect.

out the other 9 requests. Thus, while sending out multiple messages, the parent process often has to wait through multiple context switches, as the child processes receive and act upon their messages. In contrast, in a non-idle system, most messages from the parent to the child processes are not acted upon immediately by the children (since they are usually in waiting for a previous I/O). As a result, the parent can finish sending the messages to the children in a short amount of time. For example, for full stripe requests to a non-idle system, the request overhead is approximately 3.5 ms per logical request.

Request overhead also increases with the number of total disks in the system. This is due to the increased message passing overhead with more message passing channels. In general, this effect is much less pronounced on a non-idle system. However, it is true that RAID Level 1's 20 disks require more CPU power than RAID Level 0's 10 disks. By limiting the experiments to 20 disks as discussed in Section 5.2, we prevent this increased CPU demand from unfairly penalizing the throughput per disk of RAID Level 1.

8.1.3. Seek Optimization

RAID Level 1 reads are almost identical to RAID Level 0 reads. A key difference, however, is the disconnect time. The disconnect time of RAID Level 1 is shorter because RAID Level 1 requests can choose between two disks which have the same data. By choosing the copy of the data which results in the shorter seek time, the average disconnect time drops 3-4 ms from RAID Level 0. [Bitton 88]

8.1.4. RAID Level 5 Individual Writes

The RAID paper assumed that a RAID Level 5 individual write was four equal disk accesses. Because the four disk accesses are issued to two disks, each disk sees a pair of requests: first a read of the current data or parity, then a write of the new data or parity. The lifetime shown in Figure 7 focuses on one of these disks, say the data disk. Note the two distinct disk accesses in the broken out trace of RAID Level 5 individual writes in Figure 7. For the first disk access (the read), we need to add synchronization because we are using two independent disks. However, RAID Level 5 individual writes are unique in that two disks that cooperate in one request do not necessarily cooperate in the next request. Thus, they could have different current head positions. Because of this, we need to synchronize not only different rotations, but also different seek distances. Just as we synchronized multiple rotations by taking their maximum, we synchronize multiple *seek + rotations* by taking the maximum of a number of random variables, each distributed as a *seek + rotation*. This adds approximately 3.6 ms to an average seek plus an average rotation.

For the second disk access (the write), no seek is needed. Because the disk has just finished reading the old data, the new data can be written without moving the disk head. However, we do see a full rotation instead of an average rotation. Thus, we save an average seek (15 ms) but pay an extra half rotation (8.3 ms). We refer to this as RAID Level 5 *saving a seek*.

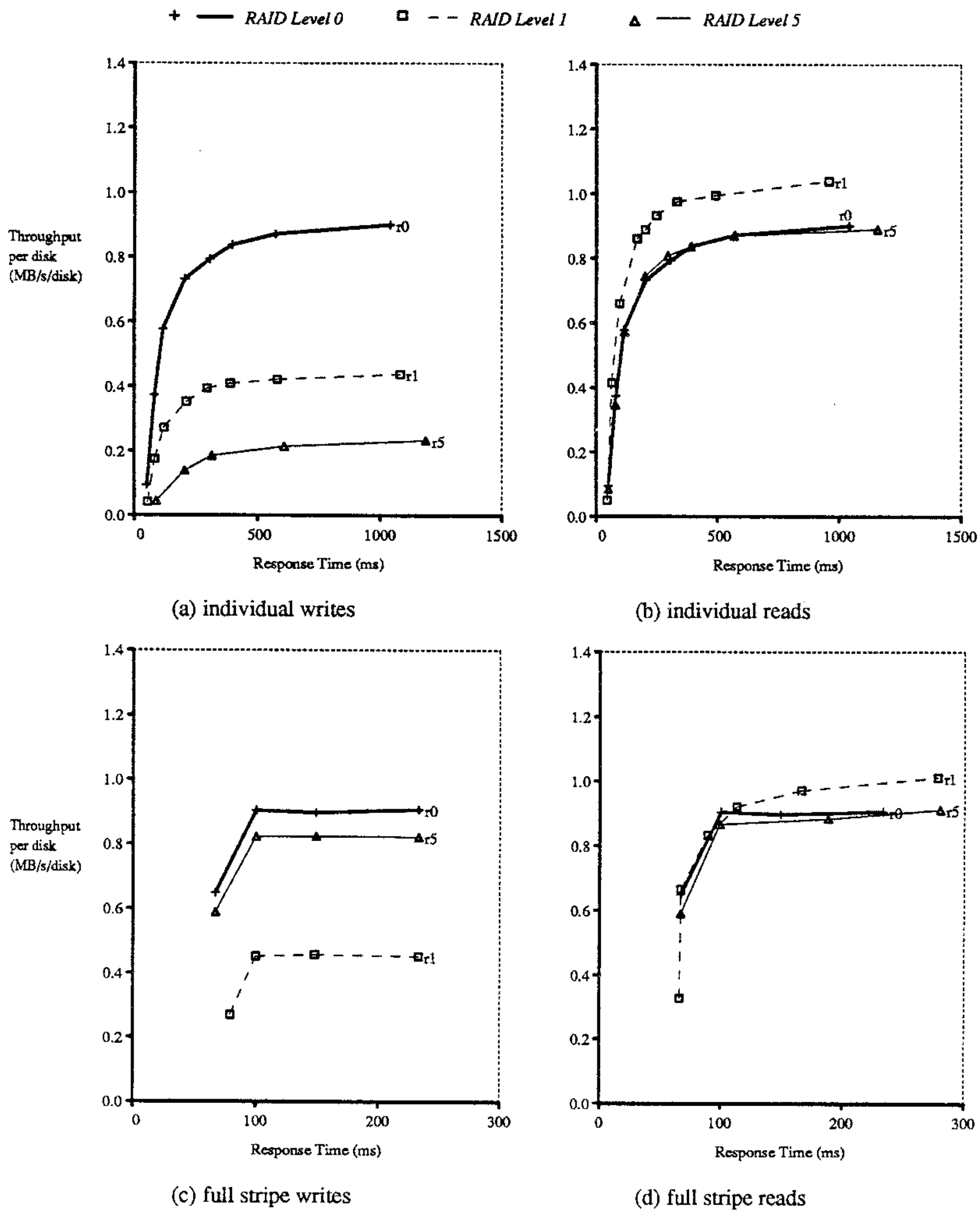
8.2. Analyzing a Saturated System

As discussed in Section 6.5, we control the system load by controlling the number of outstanding logical request in the system. Increasing the target response time allows the system to have more outstanding logical requests at a time. This causes higher disk utilization and correspondingly higher throughput. By varying the load, we can graph absolute throughput per disk versus response time. Figures 8 show these graphs for the RAID paper workloads: large reads, large writes, small reads, and small writes. As before, large means a full stripe; small means an individual request.

Note that the minimum response times are those discussed in Section 8.1. As expected, when we allow requests to have longer service times, we see higher throughput. Eventually, when the disks are fully utilized, increasing the response time no longer increases the throughput. The RAID paper analysis deals with this point of *maximum throughput*. By presenting RAID Levels 1 and 5's maximum throughput per disk relative to RAID Level 0's, we can make a direct comparison to the performance predicted in the RAID paper (Figure 9).

Figure 9 shows that, for most workloads, the simple model in the RAID paper accurately predicts the actual performance of real hardware. However, there are significant differences.

In RAID Level 1, relative throughput for reads is higher than predicted by our simple model. Recall that the simple model assumed all disk accesses took a constant amount of time. Because of seek optimization, this assumption is no longer valid. We adjust the simple model by defining relative throughput as the total disk-time used by a RAID Level 0 logical request divided by the total disk-time used by the RAID Level in question. When seen in this light, we can easily adjust for different access times. Due to seek optimization, the disk-time for RAID Level 1 reads is 4 ms per disk less than RAID Level 0. This



(a) individual writes

(b) individual reads

(c) full stripe writes

(d) full stripe reads

Figure 8: Throughput vs. Response Time. Throughput is graphed as a function of response time target for 4 types of I/O requests: individual reads and writes, full stripe reads and writes. These graphs were generated by keeping a fixed number of logical requests in the queue and measuring the resulting throughput and response time. The data distribution used here is uniform aligned.

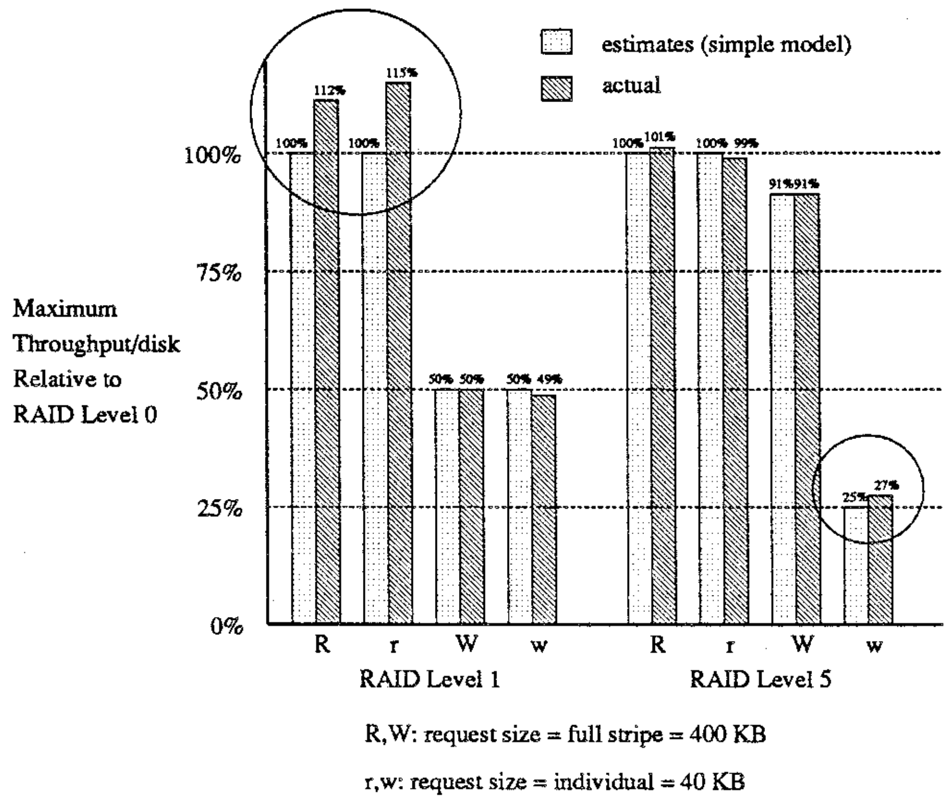


Figure 9: Comparing Maximum Throughput Against Simple Model. The measured maximum throughput per disk relative to RAID Level 0 is compared against the estimates made in the RAID paper [Patterson 88]. The only significant differences occur for RAID Level 1 reads, due to seek optimization, and RAID Level 5 individual writes, due to saving a seek.

represents approximately 10% of the total disk-time, so the adjusted simple model predicts a relative throughput of 110% for RAID Level 1. The measured relative throughput is close to this prediction, ranging from 112% - 115%.

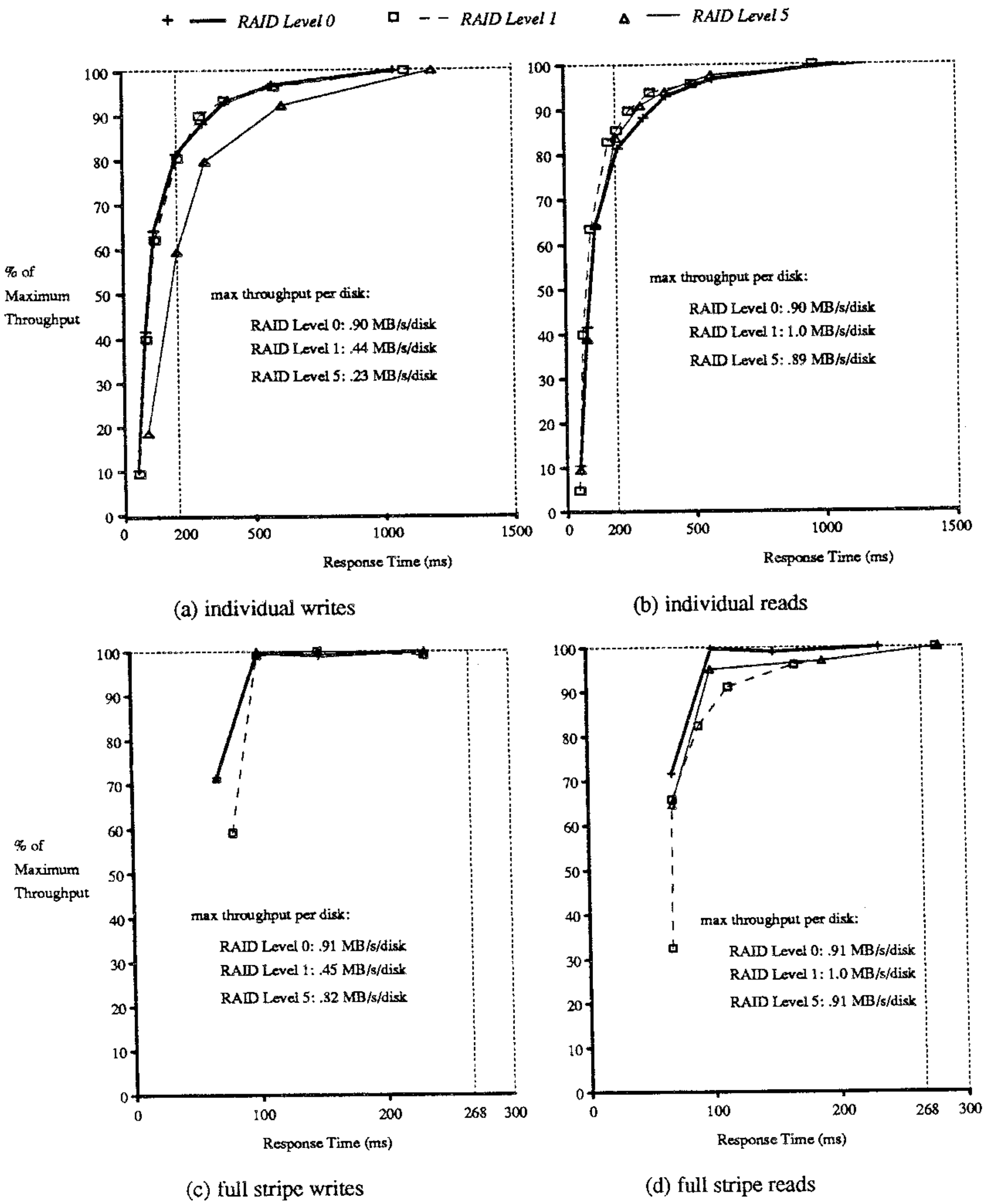
A RAID Level 5 small write causes two disks to perform a seek, an average rotation, a full track transfer, a full rotation, and a second full track transfer (total 72 ms). Applying the adjusted model to RAID Level 5 small writes, a the total disk-time is approximately 72*2 = 144 disk-ms. The total disk-time for RAID Level 0 small writes is approximately 40. Thus, while the simple model predicts a relative throughput of 25%, the adjusted simple model predicts a relative throughput of 40/144 = 28%. This agrees with the measured performance.

8.3. Maintaining Equal Response Times

Section 8.2 compared the maximum throughputs of different RAID Levels. However, the different RAID Levels reach maximum throughput at different response times. It is unfair to compare RAID Levels with different target response times. By forcing all RAID Levels to have equivalent response times, we generate a fairer comparison. Figure 8 showed that throughput for all RAID Levels drops as the response time target decreases. To better understand how throughput changes for each RAID Level, we graph throughput per disk versus response time in a slightly different way. Instead of absolute throughput per disk, we graph the *percentage of each RAID Level's maximum throughput.* For example, since RAID Level 0's maximum throughput per disk is .906 MB/s/disk for small writes, RAID Level 0 throughput is graphed as a percentage of .906 MB/s/disk in Figures 10a. RAID Level 1's maximum throughput per disk for small writes is .438 MB/s, so throughput per disk is graphed as a percentage of .438 MB/s in Figure 10a.

These response time behavior figures show us what to expect when we maintain equal response times for each RAID Level. For example, in Figure 10a, we see that at any response time less than 1000 ms, RAID Level 5 achieves a lower percentage of its maximum throughput than RAID Level 0. This is due to RAID Level 5 small writes having long response times as discussed in Section 8.1.4. Thus, RAID Level 5's relative throughput (relative to RAID Level 0) for small writes will decrease. In contrast, RAID Level 1 small writes track RAID Level 0 small writes very closely at all response times. Thus we expect RAID Level 1 small writes to maintain the same relative throughput.

In the remainder of this paper, we choose one specific response time for each workload. Our method for choosing that response time is somewhat arbitrary. We first measure the minimum response time of a RAID Level 0 (as in the idle system in Section 8.1) running a particular workload. The target response time for that workload is then set at four times that minimum response time. To summarize, if 90% of all requests on an idle RAID Level 0 return in time t, we control the response time such that 90% of all requests return in time $4*t$. For full stripe requests, target response time is 268 ms; for individual requests, target response time is 200 ms. This is intended to allow some freedom to queue requests in order to achieve higher throughput, but not to allow queues to grow too deep. Notice that four times the idle RAID



(a) individual writes

(b) individual reads

(c) full stripe writes

(d) full stripe reads

Figure 10: Percentage of Maximum Throughput. Throughput per disk for each RAID Level is shown as a percentage of the maximum throughput per disk for that RAID Level. Thus, we can see the relative effects of changing the response time target. For example, in Figure 10a, RAID Level 5 achieves a lower percentage of its maximum throughput than either RAID Level 0 or 1. Data distribution is uniform aligned.

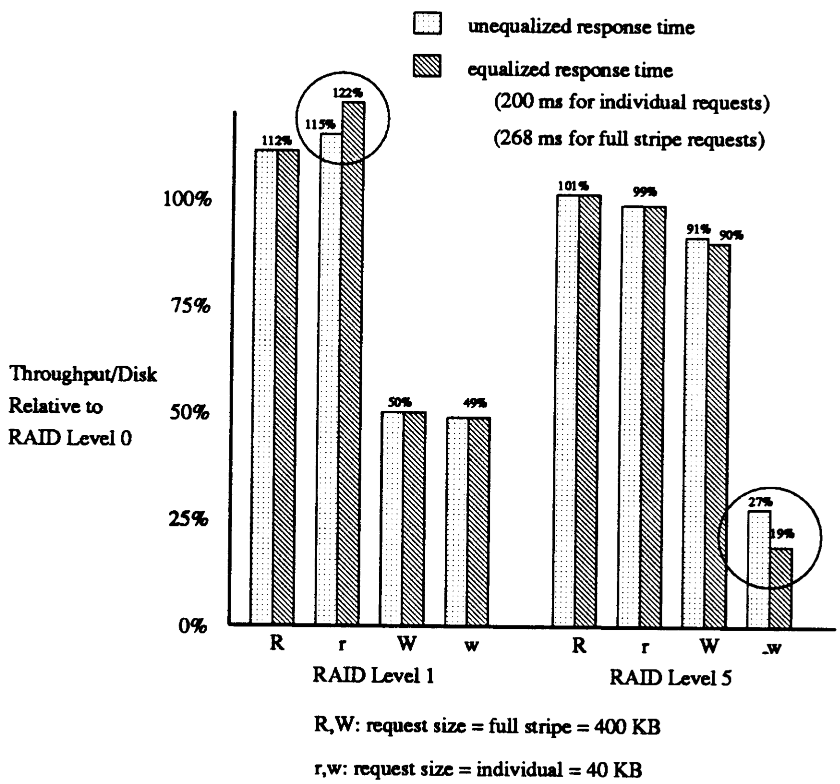


Figure 11: Before and After Equalizing Response Times. Throughput per disk relative to RAID Level 0 is shown before and after maintaining equal response times. We see that equalizing to our target response times affects relative throughput for RAID Level 1 individual reads and RAID Level 5 individual writes.

Level 0 response time for large requests easily exceeds the response time needed to achieve maximum throughput. An interesting future experiment would be to restrict large requests to a small percent over the minimum response time.

Figure 11 shows the relative throughput per disk of RAID Levels 1 and 5 before and after equalizing to these target response times. Note that at the particular response times we chose, only two workloads show significant change relative to RAID Level 0: RAID Level 5 small writes and RAID Level 1 small reads. We discussed previously how the throughput for RAID Level 5 small writes changed as we maintained equal response times. Similarly, Figure 10b shows that, at our 200 ms target response time, RAID Level 1 has is at a slightly higher percentage of its maximum throughput.

This section has focused on response times. We have seen that limiting response times causes less than maximum throughput for all RAID Levels. In general, RAID Levels with longer response times are penalized more than RAID Levels with shorter response times.

9. More Realistic Workloads

In Section 8, we analyzed the RAID paper workload of full stripe or individual requests, 100% reads or 100% writes. We started by analyzing idle systems (minimum response time) and saturated systems (maximum throughput). We finished by analyzing systems with equivalent response times. While still maintaining equivalent response times, we now begin modifying the actual workload: request size, data distribution, and read/write ratio.

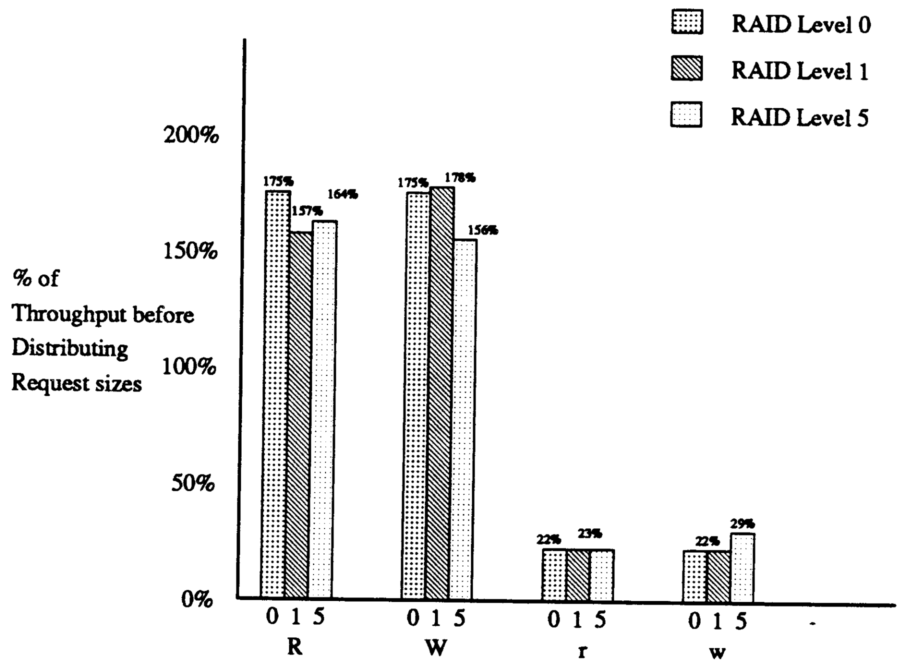
9.1. Distributing Request Sizes

A key characteristic of any workload is the logical requests sizes. Until now, large requests have been full stripe requests and small requests have been individual requests. Thus, with 10 disks and track-level striping, large requests have been exactly 400 KB and individual requests have been exactly 40 KB. However, in real-world systems, large requests are often much larger than 400 KB [Bucher 80] and small requests are often much smaller than 40 KB [Ousterhout 85, Anon 85]. Also, applications rarely issue requests that are all the same size. Rather, they issue requests of various sizes. We therefore make two changes to the request size distribution:

(1) We no longer restrict the workloads to one particular size. Rather, we use a *distribution* of request sizes. For large requests, we generate request sizes derived from a normal distribution; for small requests, we generate request sizes based on an exponential distribution.

(2) We change the average size of both large and small requests: an average large request changes from 400 KB to 1.5 MB (approximately 4 stripes); an average small request changes from 40 KB to 6 KB, the closest we could come to one 4KB sector.

Thus, the large requests get larger and the small requests get smaller. This significant change will alter many of our results; qualitatively, however, what we have learned so far will still hold. Seek optimization will continue to benefit RAID Level 1 reads; RAID Level 5 small writes will still save a seek. Changes to our target response times will follow the changes in RAID Level 0 minimum response times



Before distributing request sizes
R,W: request size = full stripe = 400 KB
r,w: request size = individual = 40 KB

After distributing request sizes
R,W: request size = normal distribution, mean 1.5 MB
r,w: request size = exponential, mean 6 KB

Figure 12: Effect on Absolute Throughput Of Changing Request Sizes. Throughput after distributing request sizes is shown as a percentage of the throughput before distributing request sizes. Throughput for large requests improved to about 160%; throughput for small requests decreased to about 25% of the undistributed request sizes.

(new response time target for large requests will be 780 ms; new response time target for small requests will be 148 ms).

Because large requests using our new size distributions will usually cover multiple stripes, each disk transfers more information per seek than before and absolute throughput will increase for all RAID Levels. In contrast, because small requests are smaller on average, less data is transferred per seek and absolute throughput for small requests will decrease for all RAID Levels. These trends in absolute throughput are shown in Figure 12, which shows throughput after distributing request sizes as a percentage of the throughput before distributing request sizes.

The same general trends hold for all RAID Levels: throughput increases for large requests and decreases for small requests. However, there is some variation in how much each RAID Level changes. To better picture how the performance of RAID Levels 1 and 5 change relative to RAID Level 0, we plot the new relative throughput per disk. We show the relative throughput before and after we distribute and change the means of the request sizes. The following four subsections discuss these results, shown in Figure 13.

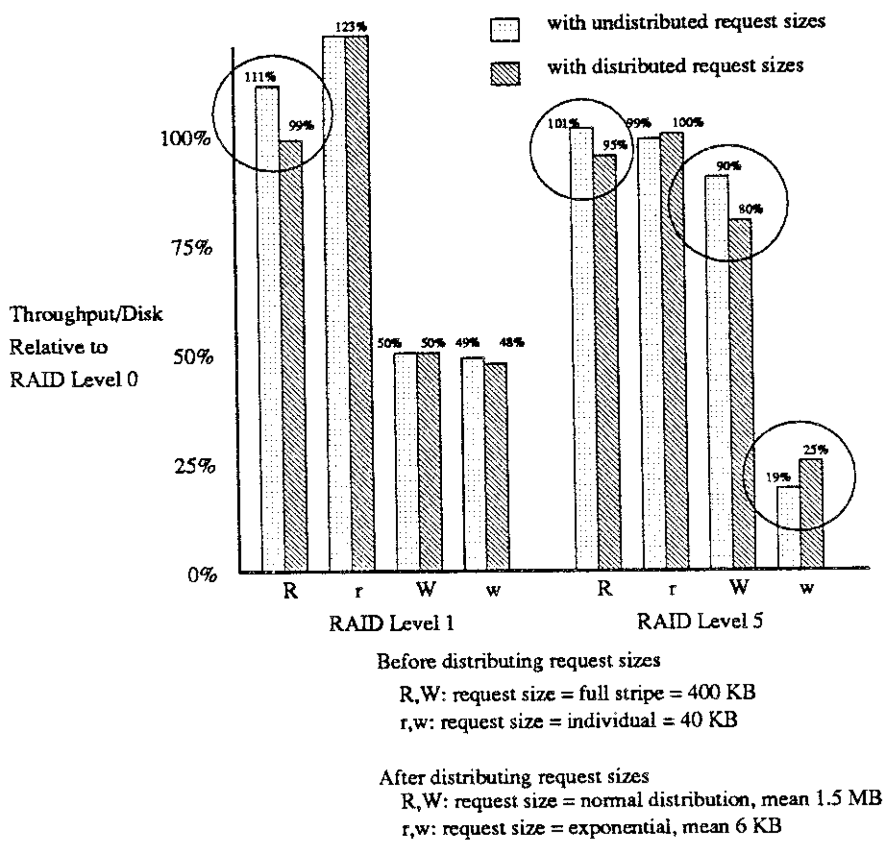


Figure 13: Relative Throughput Before and After Changing Request Sizes. In this figure, we see how the throughput of RAID Levels 1 and 5 change relative to RAID Level 0 as request sizes are distributed. Distributing the request sizes decreased the relative throughput per disk for RAID Level 1 large reads due to the lessening importance of seek optimization. RAID Level 5 large reads decreased in relative throughput due to reading the parity tracks. RAID Level 5 large writes decreased in relative throughput due to the presence of partial stripe writes. RAID Level 5 small writes increased in relative throughput due to a different response time target. Data distribution is uniform aligned.

9.1.1. RAID Level 1

With larger requests, each disk access takes longer. Because of this, savings due to seek optimization are a smaller fraction of the access time of each disk. Thus, RAID Level 1 large reads, which benefit from seek optimization, show less performance advantage over RAID Level 0. However, we do expect RAID Level 1 to still have slightly higher throughput per disk than RAID Level 0. Unfortunately, the request overhead for RAID Level 1 is 3-4 ms higher than for RAID Level 0. This cancels out the slight performance advantage of seek optimization, and makes RAID Level 1 throughput for large reads equal to RAID Level 0.

With the other workloads, large writes, small reads, and small writes, the relative throughput of RAID Level 1 turns out to be the same.

9.1.2. RAID Level 5 Small Writes

As shown in Figure 13, RAID Level 5 small writes improve in relative throughput per disk. Two factors combine to cause this improvement:

First, by saving a seek on the data and parity write, we save a fixed amount of time. With smaller requests, the total request time is shorter and this seek savings is a slightly larger fraction of the entire request time. This larger seek savings accounts for 1% of the 6% change we see in Figure 13.

Most of the improved relative throughput is due to selecting a response time target which is more favorable for RAID Level 5 small writes than the response time target for undistributed request sizes. Figure 14 is similar to Figure 10a characterizing the throughput/response-time profile for small writes. In Figure 10a, throughput at the target response time (200 ms) was at 59% of maximum throughput for RAID Level 5 and 81% for RAID Level 0. In Figure 14, throughput at the target response time (148 ms) remains roughly the same for RAID Level 0 (85%); However, RAID Level 5 at this response time is closer to its maximum throughput (72%) than before. As a result, RAID Level 5's relative throughput is higher than in Section 8.3.

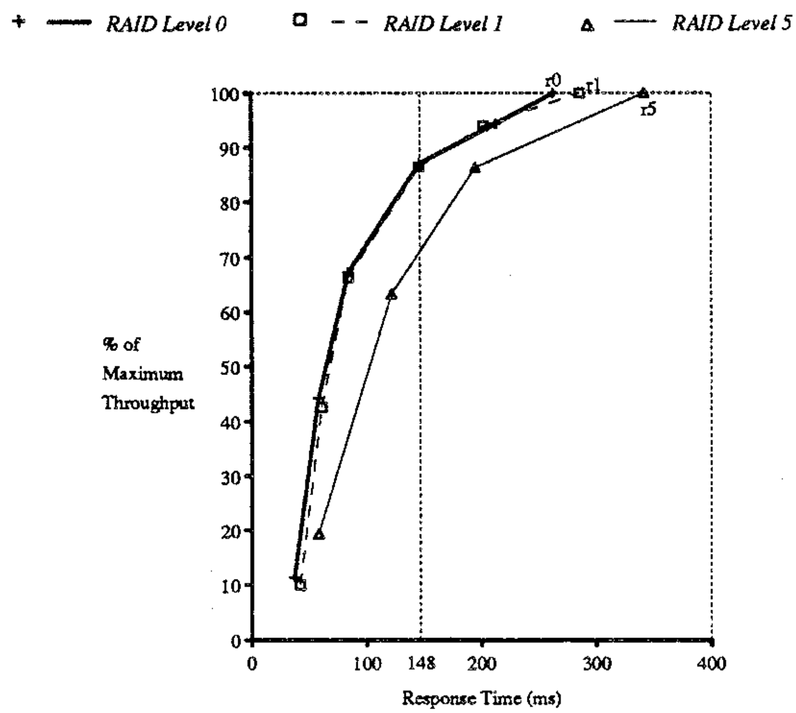


Figure 14: Percent of Maximum Throughput for Small Writes. Similar to Figure 10, throughput per disk for each RAID Level is shown as a percentage of the maximum throughput per disk for that RAID Level. Size distribution is exponential with a mean of 6 KB. Note that the target response time of 148 ms is more favorable for RAID Level 5 than the target response time in Figure 10a. Data distribution is uniform aligned.

9.1.3. RAID Level 5 Large Reads

For full stripe reads, the relative throughput of RAID Level 5 was 100%. This result was expected, as no extra redundant information needed to be read in, and the number of disk accesses in the physical request equaled the number of disk accesses in the logical request. However, now that requests cover multiple stripes, we can no longer simply read the data and ignore parity. For example, in Figure 1b, if the logical request reads logical tracks 0-13, disk 1 must read its physical tracks 0, 2, and 3. Because the experiment is run in user-level, we do not have control of the channel program. Rather, because physical tracks 0, 2, and 3 are not contiguous, we must issue two separate I/O's to disk 1. First, read physical track 0. Second, read physical track 2-3. Unfortunately, by the time the child process is able to complete the first I/O and issue the second I/O, the disk has rotated enough to miss the start of track 2. Thus, issuing two

I/O's in order to skip over the parity track costs a full rotation, the same as simply reading the parity track and issuing one large I/O.

An average size (4 full stripes, or 40 tracks) request will need to skip over two parity tracks. The overhead, then, is 2/40 or 5%. This accounts for the 5% drop in throughput shown in Figure 13.

With tighter control of the channel program, we could build one channel program to issue multiple I/O's. This would save on turnaround time between the two I/O's and prevent the missed revolution. Another solution is to map the data such that the first data sector on a track after a parity track starts several sectors past the previous data sector.

9.1.4. RAID Level 5 Large Writes

For writes which exactly cover a number of full stripes, performance is straightforward. To maintain the parity information, one parity track must be written for every 10 data tracks. Full stripe writes do not need to read any information. However, when request sizes become distributed, most requests will not cover an exact number of full stripes. Usually, one or both ends of the request will be a partial stripe. For example, in Figure 1b, if a logical request covers logical tracks 0-13, stripes 0, 1, and 2 are full stripe writes but stripe 3 is only *partially* written (logical tracks 12 and 13). Thus, although stripes 0-2 act as full stripe writes with relative throughput $\frac{N-1}{N}$, stripe 3 acts as a partial stripe write. As discussed in Section 6.4, partial stripe writes can have relative performance ranging from 25% to $\frac{N-2}{N}$.

In this section, we allow at most one partial stripe per request. This is done by using the *uniform aligned* data distribution (see Section 6.6). With this data distribution, requests larger than one full stripe are forced to either begin or end at a full stripe boundary. Introducing one partial stripe per request causes the relative throughput of RAID Level 5 large writes to drop from 90% to 80%. When we use data distributions other than uniform aligned (as in the following section), we see up to two partial stripes per request, and a correspondingly higher drop in relative throughput.

9.2. Varying the Data Distribution

Because we interleave data across disks in fixed units, hot spots tend to be spread among several disks and naturally smoothed out. Realistically, however, some disks will still receive more requests than

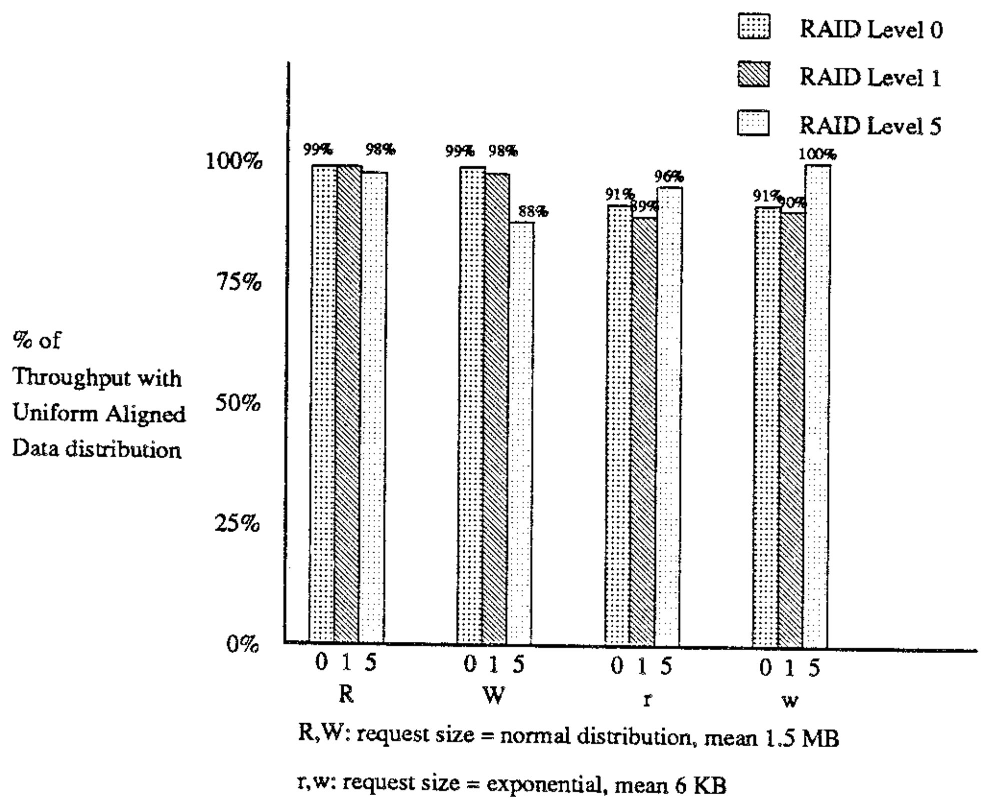


Figure 15: Change in Absolute Throughput After Skewing Data Distribution. Throughput after skewing the data distribution is shown as a percentage of the throughput before skewing the data distribution. Note that skewing the data distribution has little effect on throughput for large requests, whereas throughput for small requests decreases by approximately 10%.

others. In this section, we remove the assumption of uniform disk utilization by using a *skewed* data distribution. We continue to maintain equal response times. We also continue to use the most recent definitions of large and small requests (large is defined to be a normal distribution of request sizes with a mean of 1.5 MB; small is defined to be an exponential distribution of request sizes with a mean of 6 KB).

As discussed in Section 6.6, skewing the data distribution reduces to choosing the starting location (starting logical disk and starting stripe) of the logical request. The starting logical disk and the starting stripe on that disk are chosen independently. In a skewed distribution, the starting disk is chosen according to a normal, with standard deviation equal to the number of data disks (Figure 6). The starting stripe is always chosen according to a uniform distribution across all stripes.

Figure 15 shows the throughput after skewing the data distribution as a percentage of the throughput before skewing the data distribution. For all workloads, skewing the data distribution causes the absolute throughput to decrease. In most cases, the throughput for large requests only decreases a few percent. Because large requests cover multiple stripes, choosing the starting disk of the request generally has little effect on overall throughput. An exception to this is RAID Level 5 large writes (discussed below), whose throughput decreases more than RAID Levels 0 or 1. Small requests show a larger (10%) decrease in throughput. An exception is again RAID Level 5, whose throughput decreases only slightly.

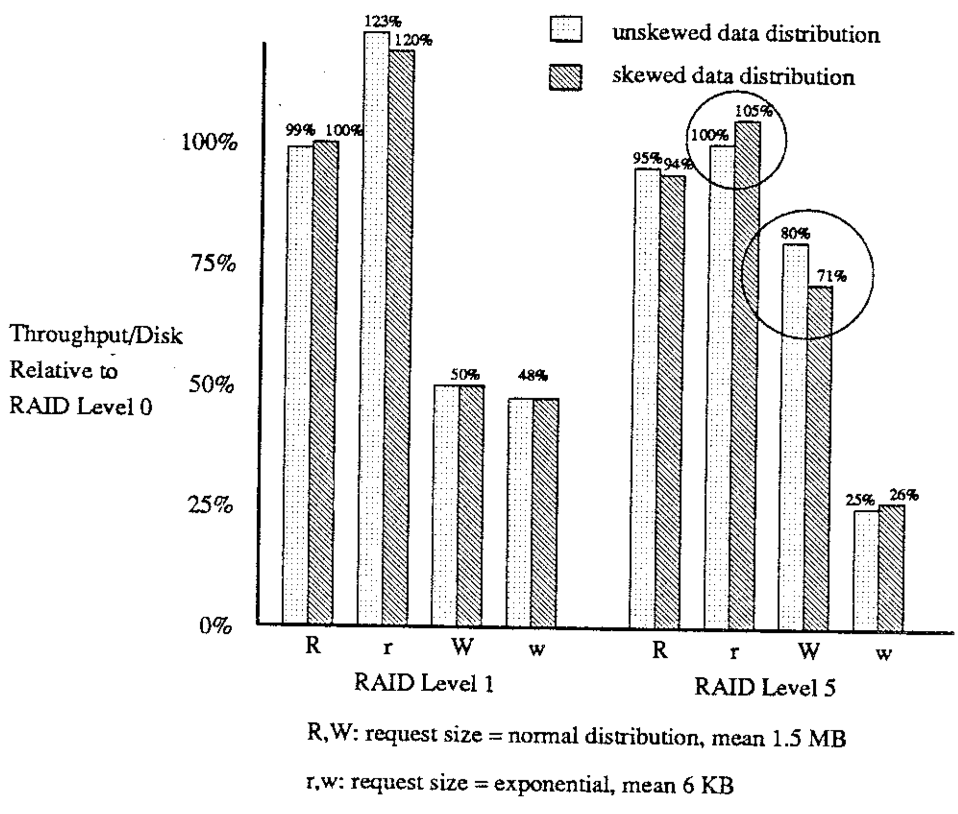


Figure 16: Relative Throughput Before and After Skewing the Data Distribution. This graph shows the effect of skewing the data distribution on relative throughput. The data distribution was changed from uniform aligned to skewed. RAID Level 5 small reads improved slightly due to the location of the parity tracks. RAID Level 5 large writes degraded due to additional partial stripe writes.

RAID Level 5 large write throughput decreases significantly because of additional partial stripe writes. When distributing the request sizes, a number of partial stripe writes were introduced. Using the uniform aligned data distribution, each request had at most one partial stripe. Now, with the skewed data distribution, no attempt is made to align the request on a full stripe boundary. Thus, up to two partial stripe writes can occur per request and throughput decreases accordingly.

For small requests, the general trend when skewing the data distribution is a decrease in throughput of 10%. However, RAID Level 5 throughput stays relatively constant. There are several reasons for this difference.

First, the skew of the data distribution is done on the logical disk. In RAID Level 0 the logical disk is identical to the physical disk. In RAID Level 1 the logical disk maps to one or both of two physical disks, independent of which stripe is accessed. However, for RAID Level 5, a logical disk does not map into a single physical disk. Rather, the mapping from logical disk to physical disk depends on which stripe is accessed. For example, in Figure 1b, logical disk 0 (stripes 0, 4, 8, etc.) is spread over physical disks 0 and 1. Thus, the skew on the data distribution is smoothed over by the logical to physical mapping done by RAID Level 5. This lessens the effect of skewing the data distribution for both RAID Level 5 small reads and small writes.

For RAID Level 5 small writes, the skewed data distribution is further smoothed because two disks are involved in each request. Recall that the starting stripe is always chosen uniformly over all stripes. Thus, although the choice of the data disk is non-uniform, the choice of the parity disk, which depends on both the starting disk and the starting stripe, is uniform over all disks. Having the parity disk uniformly chosen from all disks again smooths the skew on the disks.

We again plot the relative throughput for each RAID Level, both with and without the data distribution skewing (Figure 16).

9.3. Mixing Reads and Writes

Many different types of workloads exist in the real world. Very few, if any, are 100% reads or writes. We have used 100% reads or writes to better understand how varying other workload parameters affects performance. Now, keeping equalized response times, distributed request sizes, and skewed data distributions, we at last mix reads and writes.

At first thought, we may expect the throughput for RAID Levels 1 and 5 to be the weighted average of the throughputs for 100% reads and 100% writes. This would cause linear variation in Figure 17. Instead, though close to linear, throughput changes superlinearly with the fraction of reads in the workload. To understand why throughput is superlinear, consider an idle system receiving a stream of read or write requests. Assume reads have a response time of R and writes have a response time of W. Estimating throughput as the reciprocal of the average response time, the throughput for 100% reads is $\frac{1}{R}$ and the throughput for 100% writes is $\frac{1}{W}$. A mix of 50% reads and 50% writes will not have a throughput that is

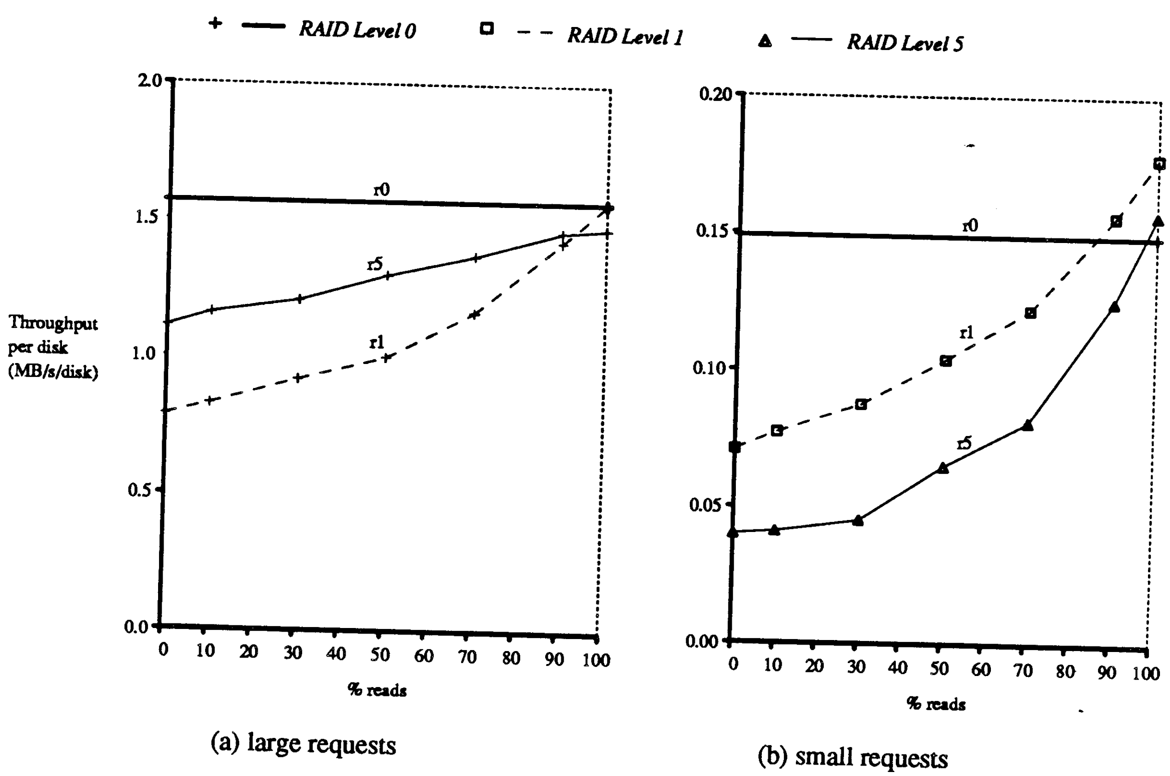


Figure 17: Effect of Mixing Reads and Writes. Throughput per disk is graphed as a function of the percentage of reads in the workload. RAID Level 0 reads and writes are assumed to be equivalent, and are shown as a horizontal line in both graphs. The size distribution for large requests is a normal with a mean of 1.5 MB (response time target of 780 ms). The size distribution for small requests is an exponential with a mean of 6 KB (response time target of 148 ms). Data distribution is skewed.

the weighted average between reads and writes, i.e. $\left[\frac{\frac{1}{R}+\frac{1}{W}}{2}\right]$. Rather, because the average response time is $\frac{R+W}{2}$, the throughput of the system should be $\frac{2}{R+W}$. Estimating throughput as the average of the constituent throughputs (averaging the reciprocal of the response times) will always be higher than the actual throughput (the reciprocal of the average response time). Thus, a graph of throughput against fraction of reads will be superlinear.

9.4. Summary of More Realistic Workloads

To summarize, we look at Figures 17. Note that for small requests, RAID Level 1 consistently yields higher throughput per disk than RAID 5. In fact, with over 90% reads, seek optimization allows RAID Level 1 to yield higher throughput per disk than even RAID Level 0.

In contrast, for large requests, RAID Level 5 almost always performs better than RAID Level 1. From 0% reads to almost 95% reads, RAID Level 5 yields higher throughput per disk than RAID Level 1.

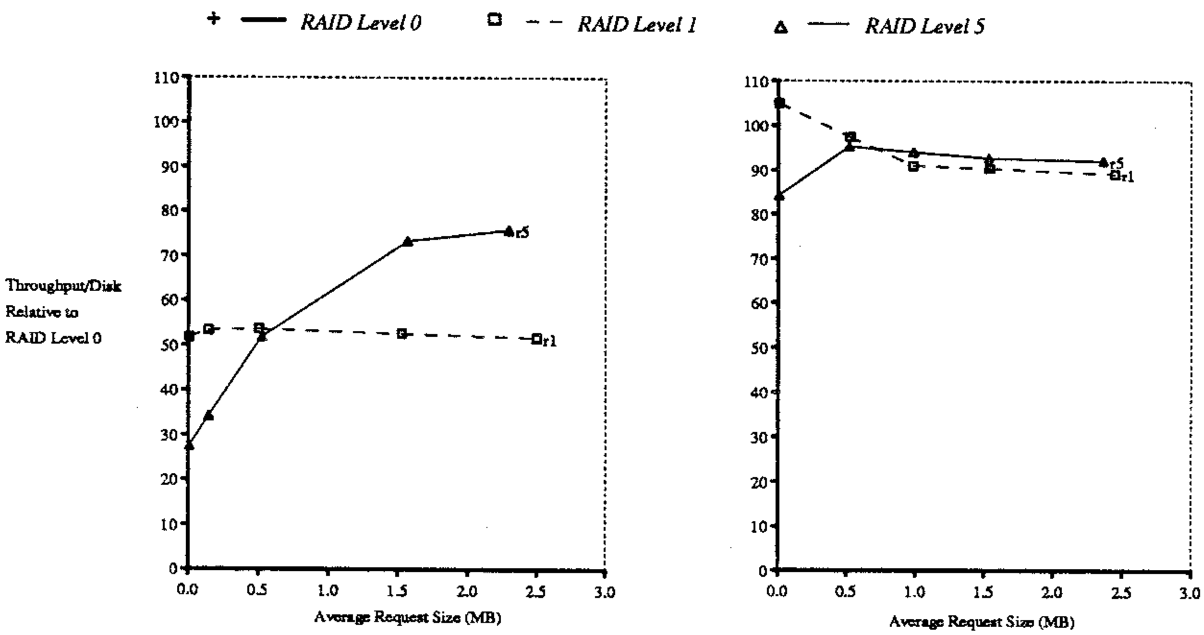
10. Additional issues

In this section, we explore additional issues, such as further varying the request size distribution, varying the unit of interleaving, and scaling the number of disks. Unless otherwise stated, we define large as a normal distribution with mean 1.5 MB and small as an exponential distribution with mean 6 KB. We also continue to equalize response times and skew the data distribution. A major change from the previous section is that we now define the read workload to be a mixture of 90% reads and 10% writes; similarly, we now define the write workload to be a mixture of 90% writes and 10% reads.

10.1. Varying the Request Sizes

In Section 9.1, we both distributed the request sizes and changed the mean. In this section, we continue to change the mean of the request size distribution by using various normal distributions.

Figure 18 graphs the relative throughput for RAID Levels 1 and 5 against average request size. In Figure 18a, RAID Level 1 relative throughput for writes is approximately 50% for all sizes, as expected. In contrast, relative throughput for RAID Level 5 writes increases as request size increases. As requests



(a) 90% writes

(b) 90% reads

Figure 18: Effect of Varying the Average Request Size. These graphs show the relative throughput per disk of RAID Levels 1 and 5 as a function of average request size. In general, the relative throughput of RAID Level 5 increases with increasing request size, whereas the relative throughput for RAID Level 1 changes very little with request size. Data distribution is skewed.

become larger and cover more full stripes, RAID Level 5 becomes dominated by full stripe write performance rather than the partial stripe write performance and relative throughput approaches $\frac{N-1}{N}$. Below .5 MB, RAID Level 1 has higher throughput per disk than RAID Level 5; at sizes larger than .5 MB, RAID Level 5 yields higher throughput per disk than RAID Level 1.

In Figure 18b, RAID Level 1 relative throughput for reads decreases with increasing request size. With small requests, seek optimization pushes RAID Level 1 relative throughput above 100%. As requests get larger and response time increases, the benefit of seek optimization is minimized. Relative throughput then drops to 90%. The 10% writes in the workload cause the total relative throughput to be less than the read performance in Section 9.2 (120% for small requests, 100% for large requests).

For RAID Level 5, relative throughput is 85% for small request sizes, then increases to 90%-95% as request sizes increase. This effect is due almost entirely to the 10% writes in the workload. For small request sizes, writes have a relative throughput of 20%-25%. However, at larger request sizes, the relative throughput of writes is 70%-80%, so relative throughput increases.

10.2. Sector Striping vs. Track Striping

Up to now, we have always used track striping. In this section, we explore sector level striping. We address two questions:

- Is it possible to analyze RAID Level 1 and 5 independent of the unit of interleaving?

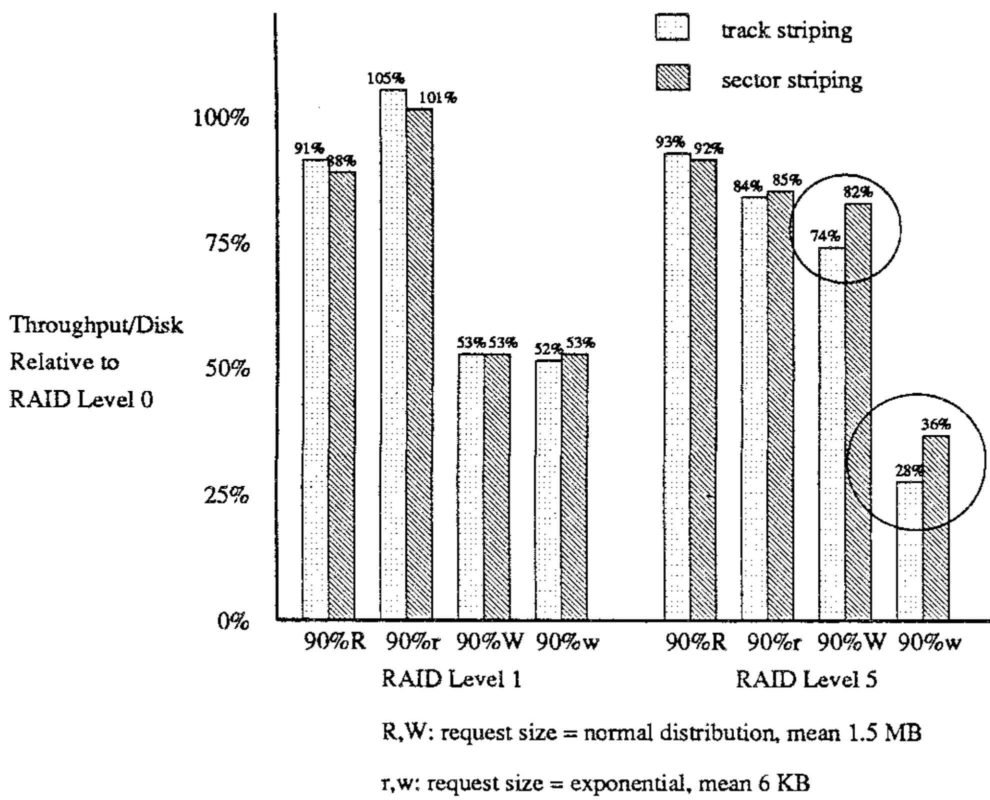
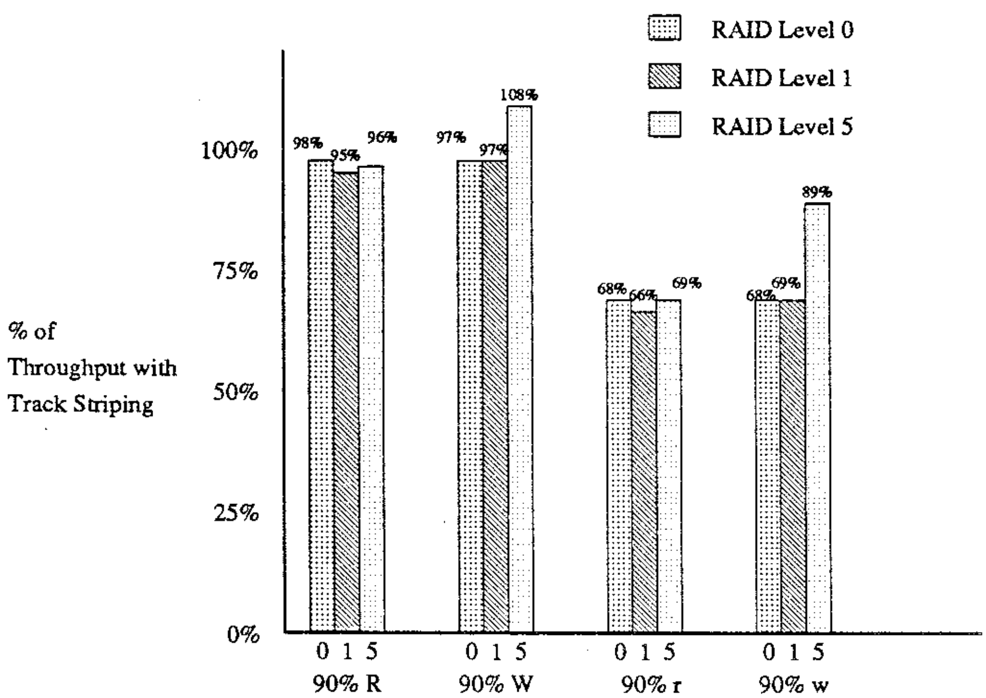


Figure 19: Relative Throughput for Track and Sector Striping. Relative throughput per disk with sector striping is compared against the relative throughput per disk with track striping. We see that using sector striping only affects relative throughput for RAID Level 5 writes. The response time target for small requests is 148 ms; the response time target for large requests is 780 ms. The read workload is 90% reads and 10% writes; the write workload is 90% writes and 10% reads. Data distribution is skewed.

- Which unit of interleaving gives the best absolute throughput?

The RAID paper analysis was independent of the unit of interleaving. Dependence from unit of interleaving, as well as from factors such as hardware specifications, was eliminated partly by evaluating the throughput relative to RAID Level 0. Also, the analysis achieved independence from the unit of interleaving by basing the analysis on full stripe or individual requests. These request sizes scaled with the unit of interleaving. In our experiment, we fix the request size distribution independent of the unit of interleaving to learn if identical workloads can be analyzed without regard for the unit of interleaving. Figure 19 shows the relative throughput for RAID Levels 1 and 5 for various sizes with 90% writes and 90% reads.



R,W: request size = normal distribution, mean 1.5 MB

r,w: request size = exponential, mean 6 KB

Figure 20: Change in Absolute Throughput after Changing from Track to Sector Striping. Throughput with sector striping is shown as a percentage of the throughput with track striping. Throughput with track striping is generally better than throughput with sector striping. An exception to this is RAID Level 5 large writes. The reads workload is 90% reads and 10% writes; the write workload is 90% writes and 10% reads. Data distribution is skewed.

For RAID Level 1, relative throughput with sector striping is close to relative throughput with track striping. Similarly, for RAID Level 5 reads, relative throughput with sector striping closely matches relative throughput with sector striping. However, for RAID Level 5 writes, the relative throughput changes with the unit of interleaving. Thus, although analysis for RAID Level 1 and RAID Level 5 reads can be done with little regard to unit of interleaving, the analysis for RAID Level 5 writes should take into account the unit of interleaving.

To compare sector striping to track striping in terms of absolute throughput, we show throughput with sector striping as a percentage of throughput with track striping (Figure 20). The general trend in Figure 20 is that changing from track to sector striping causes throughput to decrease. This decrease is more pronounced for small requests than for large requests. For very small requests (single sector), the disks see the same stream of physical requests using either sector striping or track striping. Slightly larger requests (larger than a sector but less than a track) are mapped onto multiple disks with sector striping but onto a single disk with track striping. Thus, in track striping, the single disk is transferring more data per seek than the multiple disks are in sector striping. This causes lower throughput with sector striping. With extremely large requests (more than 10 tracks), both sector striping and track striping cause disks to transfer the same amount of information, leading to roughly the same throughput.

The sole exception to this trend in changing from track to sector striping is RAID Level 5 writes. The trend of decreased throughput with sector striping was caused by spreading a request over more disks, with each disk transferring less data. However, RAID Level 5 writes benefit by using more disks in servicing a request. Relative throughput for partial stripe writes ranges from 25% for single disk partial stripe writes to $\frac{N-2}{N}$ for wider (many disks) partial stripe writes. Thus, spreading requests over more disks leads to wider partial stripes and higher relative throughput. This can offset the performance degradation caused by having each disk transfer less data. With very large requests, throughput also increases because the partial stripe portion of the request will be a lesser fraction of the total request.

10.3. Scaling the Number of Disks

So far in this paper, we have limited the maximum number of total disks to 20. All RAID Levels have used 10 data disks, with RAID Level 1 using 20 total disks and RAID Level 5 using 11 total disks. In this section, we explore what happens as we vary the number of data disks used.

Figures scale shows how the throughput per disk varies as we vary the number of data disks. We continue to use a target response time of four times the response time of an idle RAID Level 0 system. We also continue to define small as an exponential distribution with mean 6 KB. However, we change our definition of a large request. With 10 data disks, our previous definition of a large request (normal with mean 1.5 MB) covered an average of 4 stripes. Thus, each disk transferred 4 tracks per seek. However, with more data disks, this average sized request no longer covers 4 stripes. To compensate, we scale the size of an average request along with the number of disks. We maintain an average request size of 150 KB/data disk. For example, with 20 data disks, we use an average request size of 3 MB. With this modification, an average large request will continue to cause each disk to transfer approximately 4 tracks.

In general, throughput per disk is approximately constant, showing that it is independent of the number of data disks. An important exception occurs when the total number of disks exceeds 20. For example, a RAID Level 1 system with 15 data disk (30 total disks) shows a dramatic drop in throughput per disk. As discussed in Section 5.2, this drop in throughput per disk comes from not scaling the CPU power along with the number of disks. With more than 20 disks, the CPU begins to limit performance and maintaining constant throughput per disk becomes impossible without more CPU power.

We also see that the throughput per disk of RAID Level 5 large writes (Figure 21c) increases slightly with more data disks. This is because there is always one parity disk per system, and, with more data disks in the system, the overhead of updating this parity disk affects overall system performance less. Thus, throughput per disk increases.

10.4. Multiple Disks per Channel Path

We have, so far, connected one disk per channel path (Figure 4). When a disk is ready to transfer information, no channel conflicts are possible. With more than one disk is connected to a channel path, (Figure 22) channel conflicts are possible [Ng 88]. These channel conflicts delay the disk from transferring data and cause the disk to miss a rotation (an RPS miss). Figure 23 shows the effect of connecting multiple (one or two) disks per channel path, by showing throughput with multiple disks per channel path as a fraction of throughput with one disk per channel path.

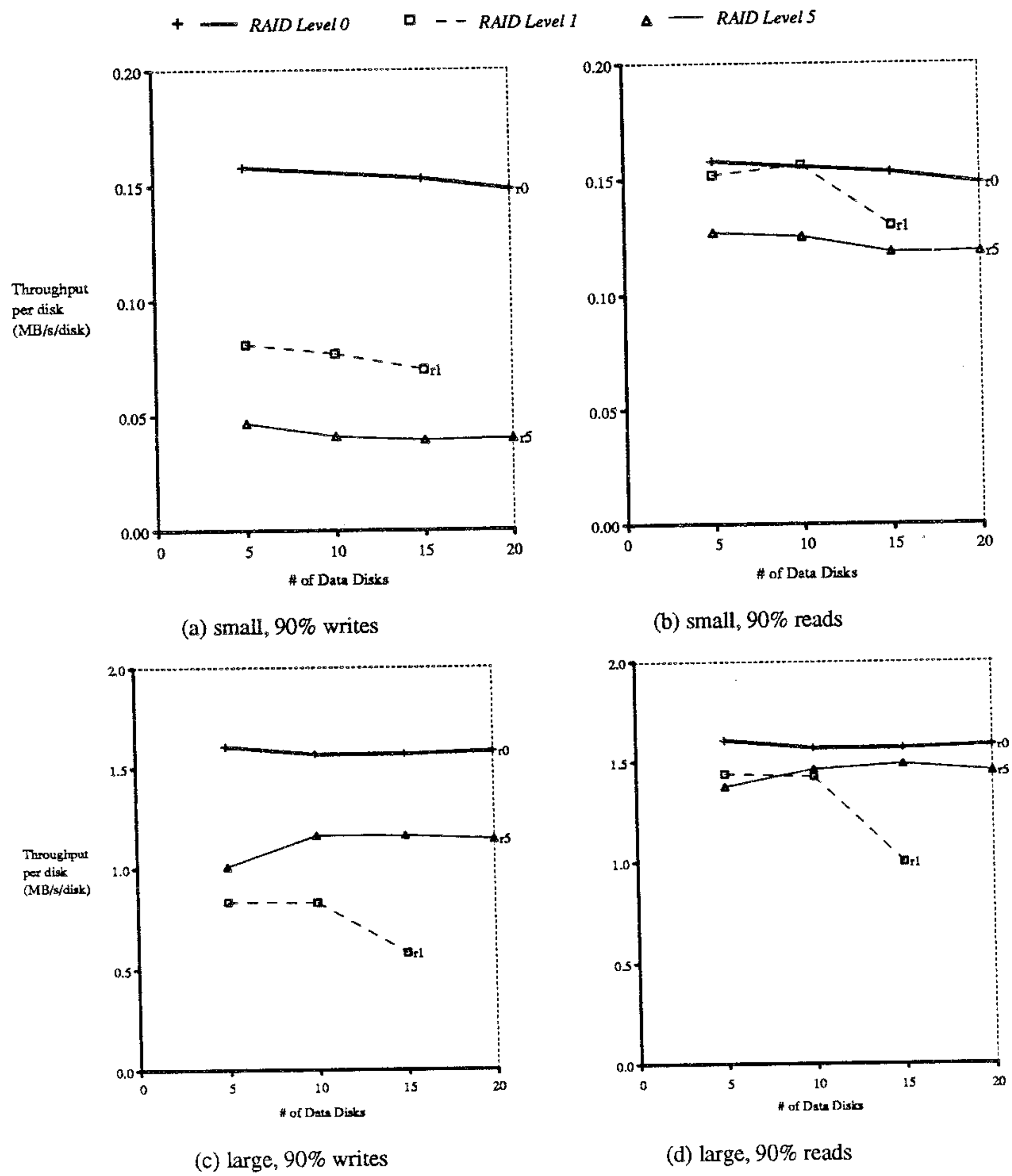


Figure 21: How RAID's Scale. Throughput per disk is shown as a function of the number of data disks. In general, throughput per disk is constant under 20 total disks. The size distribution for large requests is a normal with a mean of 150 KB per data disk. The size distribution for small requests is an exponential with a mean of 4 KB. Response time target for small requests (for all numbers of disks) is 148 ms. Response time target for large requests: 5 data disks: 720 ms, 10 data disks: 780 ms, 15 data disks: 876 ms, 20 data disks: 924 ms. Data distribution is skewed.

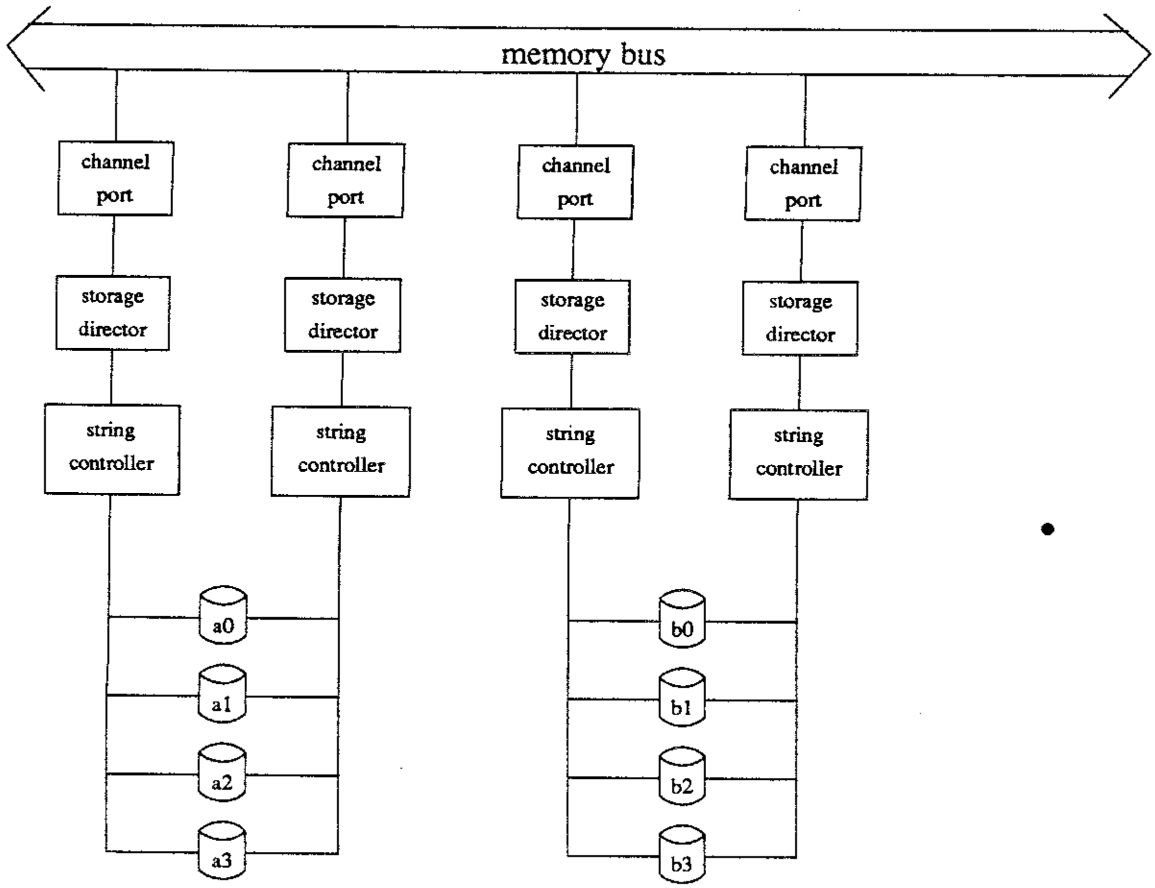


Figure 22: Architecture with Multiple Disks per Channel Path. This figure shows how multiple disks per channel are connected to the memory bus for Section 10.4. Strings of four disks are shared between two paths to the memory bus.

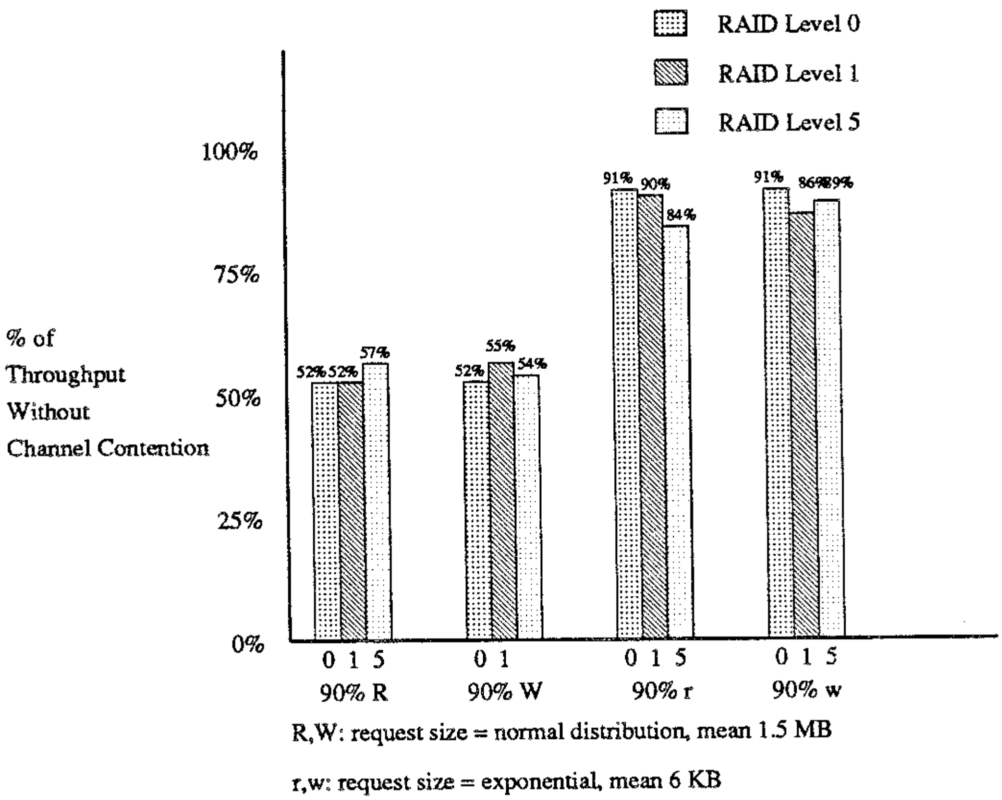


Figure 23: Change in Absolute Throughput after Connecting Multiple Disks per Channel Path. Throughput with multiple disks per channel path is shown as a percentage of throughput with one disk per channel path. Throughput for large request sizes drops to half the throughput with one disk per channel path, whereas throughput for small requests only decreases 10%.

Two channel paths are connected to one string of four disks. We represent a string by letters and the disks on a string by numbers (e.g. disks a0, a1, a2, a3 make up one string). RAID Levels 0 and 5 have a straightforward mapping: disks 0-10 are a0-a3, b0-b3, c0-c2. Thus, RAID Levels 0 and 5 have 10 or 11 disks with 6 channel paths total. RAID Level 1 has the following mapping: the primary data disks are a0-a3, b0-b3, c0-c1; the shadow disks are c2-c3, d0-d3, e0-e3.

Throughput for large requests drops to 50%-55% of the throughput with one disk per channel. Because each disk is transferring an average of 4 tracks per request, the channel is often busy. Sharing a busy channel causes a severe drop in throughput. Small requests, on the other hand, only drop in throughput by 10%-15%. Because very little time in a small request is spent transferring data (and thus tying up a channel path), little potential for channel conflicts exist and little penalty is seen.

11. Summary

We have started with a simple model for performance and, step by step, measured performance with more and more realistic experiments. We first measured maximum throughput using the RAID paper workload of 100% reads or writes, individual or full stripe requests. We found that the simple model accurately predicted performance for most cases. The main exception was RAID Level 1 reads, where seek optimization causes higher than expected throughput.

Second, we equalized response times. We found that equalizing response times hurt the relative throughput of RAID Levels with higher response times (RAID Level 5 small writes) and helped the relative throughput of RAID Levels with lower response times (RAID Level 1 small reads).

Third, we distributed the request sizes and made large requests larger and small requests smaller. We found that seek optimization ceased to noticeably help RAID Level 1 large reads. RAID Level 5 large reads were penalized for skipping parity tracks. RAID Level 5 large writes suffered from partial stripe writes. Even more partial stripe writes were generated by allowing unaligned requests.

Fourth, we mixed reads and writes. We found that RAID Level 5 had higher throughput per disk than RAID Level 1 for large requests for almost all mixes of reads and writes. In contrast, RAID Level 1 had higher throughput per disk than RAID Level 5 for small requests for all mixes reads and writes.

Lastly, we explored issues such as varying the request sizes and using sector striping. We found that track striping was usually better than sector striping, with the notable exception of RAID Level 5 large writes.

12. Future Work

We are continuing to analyze the performance of disk arrays. In particular, we are interested in arrays of small disks. We are designing, building, and evaluating a disk array of 30-50 CDC Wren disk drives. One step in that evaluation will be to carry out experiments similar to the ones in this paper. Further work will entail building a file system and running real world benchmarks on that system.

13. Acknowledgements

First and foremost, "*thanks be to God, who always leads us in His triumph in Christ. (2 Cor 2:14)*". Also thanks to Donna, Andy, and the rest of Crusade for their constant encouragement.

Technically, many of the research ideas are due to Garth Gibson, who spent many hours discussing, advising, and mothering throughout. Dave Patterson, Susan Eggers, and Alan Smith also provided much needed advice. Thanks also go to Randy Katz and Dick Wilmot for their helpful reviews. At Amdahl, John Colgrove, Art Casabal, and Romeo Narciso were indispensable resources.

The Amdahl Corporation generously provided many hours of expensive standalone time. I was supported in part by an ONR fellowship. The RAID project is funded in part by the National Science Foundation under grant #MIP-8715235. The RAID project would also like to thank Sun Microsystems Incorporated, Impress/CDC, and other industrial partners for their support.

14. Bibliography

[Amdahl 5890] ''*5890 Processors*'', Amdahl marketing publications MM001388, 1988.

[Amdahl 6380] ''*6380E Direct Access Storage Device*'', Amdahl marketing publications MM001352, 1987.

[Amdahl 67] G. Amdahl, ''*Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities*,'' Proceedings AFIPS 1967 Spring Joint Computer Conference, Vol. 30 (Atlantic City, NJ, April 1967), pp. 483-485.

[Anon 85] Anon Et Al, ''*A Measure of Transaction Processing Power*,'' Tandem Technical Report TR85.2, 1985.

[Bell 84] C.G. Bell, ''*The Mini and Micro Industries*,'' IEEE Computer, Vol. 17, No. 10 (October 1984), pp. 14-30.

[Bitton 88] D. Bitton and J. Gray, ''*Disk Shadowing*,'' in press, 1988.

[Bucher 80] I. Bucher, A. Hayes, ''*I/O Performance Measurement on CRAY-1 and CDC 7600 Computers*,'' 16th Meeting of Computer Performance Evaluation Users Group, Oct. 1980.

[Buzen 86] J. Buzen, A. Shum, ''*I/O Architecture in MVS/370 and MVS/XA*'', CMG Transactions, vol 54, Fall 1986, pp. 19-26

[Harker 81] J. Harker et al., ''*A Quarter Century of Disk File Innovation*,'' IBM Journal of Research and Development, Vol 25, No. 5, Sept. 1981, pp. 677-689.

[IBM 87] *IBM System/370XA Principle of Operations*, IBM publication SA22-7085-01, 2nd ed. Jan. 1987.

[Joy 85] B. Joy, presentation at ISSCC 1985 panel session, Feb. 1985.

[Kim 86] M. Kim, ''*Synchronized Disk Interleaving*,'' IEEE Trans. on Computers, Vol. C-35, no. 11, Nov. 1986.

[Kim 87] M. Kim, A. Nigam, G. Paul, ''*Disk Interleaving and Very Large Fast Fourier Transforms*,'' International Journal of Supercomputer Applications, Vol 1, No. 3, Fall 1987, pp. 75-96.

[Kurzweil 88] F. Kurzweil, ''*Small Disk Arrays -- The Emerging Approach to High Performance*,'' presentation at Spring COMPCON 88, March 1, 1988, San Francisco, CA.

[Moore 75] G. Moore, ''*Progress in Digital Integrated Electronics*,'' Proc. IEEE Digital Integrated Electronic Device Meeting, 1975, p. 11.

[Myers 86] W. Myers, ''*The Competitiveness of the United States Disk Industry*,'' IEEE Computer, Vol. 19, No. 11, January 1986, pp. 85-90.

[Ng 88] The 15th Annual International Symposium on Computer Architecture (SIGARCH 88), Computer Architecture News, Volume 16, Number 2, May 1988.

[Ousterhout 85] J. Ousterhout, H. Da Costa, D. Harrison, J. Kunze, M. Kupfer, J. Thompson, *"A Trace-Driven Analysis of the UNIX 4.2 BSD File System,"* ACM Operating Systems Review, Vol. 19, No. 5, Proceedings of the 10th ACM Symposium on Operating System Principles, Dec. 1-4, 1985.

[Patterson 88] D. Patterson, G. Gibson, and R. Katz, *"A Case for Redundant Arrays of Inexpensive Disks (RAID),"* ACM SIGMOD conference proceedings, Chicago, IL., June 1-3, 1988, pp. 109-116. (Also appeared as Technical Report UCB/CSD 87/391, December 1987).

[Salem 86] K. Salem, H. Garcia-Molina, *"Disk Striping,"* IEEE 1986 Int. Conf on Data Engineering, 1986.

[Schulze 88] M. Schulze, G. Gibson, R. Katz, and D. Patterson, *"How Reliable is a RAID?,"* Spring COMPCON 89, March 1, 1989, San Francisco, CA.

[Thisquen 88] J. Thisquen, *"Seek Time Measurements"*, Amdahl Peripheral Products Division Technical Report, May 9, 1988.

[UTS 88] *UTS580 User Reference Manual*, Amdahl publication ML-142292, 1988.

```
The following is the data used in this paper:

Field 1 (eg. r0, r1, r5): program used: r0 = raid0, r1 = raid1, r5 = raid5
Field 2 (eg. 10): unit of interleaving, in 4K sectors
Field 3 (eg. t13): code for read/write ratio:
    t7 = 90% reads
    t8 = 10% reads
    t9 = 100% writes
    t10 = 30% reads
    t11 = 50% reads
    t12 = 70% reads
    t12 = 70% reads
    t13 = 100% reads
Field 4 (eg. s1): code for request size distribution
    s1 = full stripe
    s2 = individual
    s3 = normal with mean 300 sectors, standard deviation 300 sectors
    s4 = exponential with mean 1 sector
    s8 = normal with mean 450 sectors, standard deviation 450 sectors
    s11 = normal with mean 150 sectors, standard deviation 150 sectors
    s29 = normal with mean 25 sectors, standard deviation 25 sectors
    s31 = normal with mean 100 sectors, standard deviation 100 sectors
    s32 = normal with mean 500 sectors, standard deviation 500 sectors
    s41 = normal with mean 600 sectors, standard deviation 600 sectors
    s42 = normal with mean 192 sectors, standard deviation 192 sectors
Field 5 (eg. l10): code for data distribution
    l5 = normal with standard deviation 15 disks
    l6 = normal with standard deviation 10 disks
    l7 = normal with standard deviation 5 disks
    l9 = normal with standard deviation 20 disks
    l10 = uniform aligned data distribution
Field 6 (eg. lat-5, lat200): target response time (for 90% latency marks)
    lat-x = keep the queue depth fixed at x outstanding logical request
    latxxx = target the response time at xxx ms
Field 7 (eg. d10): number of disks
Field 8 (eg. ro3.338): request overhead, in ms
Field 9 (eg. la233): measured response time (90% marks)
Field 10 (eg. lap.899): measured percentage of requests which met the measured
    response time
Field 11 (eg. latp.81): measured percentage of requests which met the target
    response time
Field 12 (eg. laav214.618): measured average response time in ms
Field 13 (eg. i1398): number of I/O's
Field 14 (eg. e0): amount of CPU time spent in exclusive or-ing
Field 15 (eg. sz100): measured average size of request
Field 16 (eg. szs.3): measured standard deviation in size of requests
Field 17 (eg. ti60255.4): total time of run
Field 18 (eg. th9.06298): throughput in MB/s
Field 19 (eg. io23.2012): I/O's per second
Field 20 (eg. ch40.0411): total time per access spent by child process
Field 21 (eg. chc1.27629): cpu time spent per access by child process
Field 22 (eg. qd5.01788): average queue depth over the run
Field 23 (eg. ssch13993): total number of start subchannels
Field 24 (eg. conn16.1593): average connect time per disk access
Field 25 (eg. disc22.5913): average disconnect time per disk access
Field 26 (eg. pend.23765): average function pending time per disk access
Field 27 (eg. cpu2912.22): total cpu time
Field 28 (eg. u0): run reached stability (u1) or not (u0)

r0 10 t13 s1 l10 lat-5 d10 cr0 ro3.338 la233 lap.899142 latp0 laav214.618 i1398 e0 sz100 szs0 ti60255.4 th9.06298 io23.2012 ch40.0411 chc1.27629 qd5.01788 ssch13993 conn16.1593 di
sc22.5913 pend.23765 cpu2912.22 u0
r0 10 t13 s2 l10 lat-120 d10 cr0 ro.699658 la1040 lap.899921 latp0 laav508.411 i14067 e0 sz10 szs0 ti61022.4 th9.00477 io230.522 ch40.1164 chc1.26025 qd121.024 ssch14186 conn16.15
67 disc22.6262 pend.236519 cpu6704.87 u0
r1 10 t13 s1 l10 lat-10 d20 cr1 ro3.73798 la278 lap.898718 latp0 laav191.986 i3120 e0 sz100 szs0 ti60251.5 th20.2277 io51.7829 ch35.663 chc1.2672 qd10.0321 ssch31252 conn16.161 di
sc18.1206 pend.23331 cpu6935.06 u0
```

```
r1 10 t13 s2 110 lat-240 d20 cr1 ro.819269 la957 lap.900025 latp0 laav439.637 i32400 e0 sz10 szs0 ti60894.3 th20.784 io532.069 ch35.1774 chc1.30254 qd241.778 ssch32634 conn16.1588
 disc17.5836 pend.231686 cpu17205.2 u0
r1 10 t9 s1 110 lat-5 d20 cr1 ro6.86653 la233 lap.900358 latp0 laav214.527 i1395 e0 sz100 szs0 ti60226.8 th9.04784 io23.1625 ch40.408 chc1.18748 qd5.01792 ssch27925 conn16.1441 di
sc22.9866 pend.232784 cpu5761.84 u0
r1 10 t9 s2 110 lat-120 d20 cr1 ro1.3117 la1083 lap.89986 latp0 laav519.434 i13761 e0 sz10 szs0 ti61308 th8.76785 io224.457 ch40.3154 chc1.232 qd121.046 ssch27716 conn16.146 disc2
2.9276 pend.23198 cpu10248.5 u0
r5 10 t13 s1 110 lat-6 d11 cr5 ro3.12503 la280 lap.902084 latp0 laav232.613 i2063 e0 sz100 szs0 ti80309.8 th10.0344 io25.688 ch39.9428 chc1.22715 qd6.01745 ssch20650 conn16.1585 d
isc22.4777 pend.233939 cpu4483.73 u0
r5 10 t13 s2 110 lat-132 d11 cr5 ro.462504 la1156 lap.90004 latp0 laav502.776 i15511 e0 sz10 szs0 ti61942.8 th9.78158 io250.408 ch40.1741 chc1.25509 qd133.123 ssch15643 conn16.156
9 disc22.7096 pend.232729 cpu5083.76 u0
r5 10 t9 s1 110 lat-5 d11 cr5 ro4.00511 la233 lap.904591 latp0 laav214.888 i1394 e9851.16 sz100 szs0 ti60239.1 th9.03949 io23.1411 ch40.2587 chc1.31094 qd5.01793 ssch15359 conn16.
1568 disc22.7444 pend.236465 cpu11435.3 u0
r5 10 t9 s2 110 lat-66 d11 cr5 ro1.32368 la1720 lap.899904 latp0 laav924.554 i4238 e7368.39 sz10 szs0 ti62198.1 th2.66161 io68.1371 ch36.3098 chc1.49142 qd67.0278 ssch17114 conn16
.157 disc18.8004 pend.233734 cpu10000 u0
r0 10 t13 s3 16 lat780 d10 cr0 ro3.64628 la791 lap.899719 latp.89597 laav587.633 i1068 e0 sz378.298 szs237.106 ti100740 th15.6662 io10.6015 ch95.0774 chc4.935 qd6.33708 ssch10093
conn65.8926 disc26.0005 pend.232221 cpu2220.92 u0
r0 10 t13 s4 16 lat148 d10 cr0 ro.967385 la145 lap.899984 latp.90693 laav70.2815 i18736 e0 sz1.57851 szs.954408 ti77657.5 th1.48765 io241.265 ch27.7283 chc.750901 qd17.5859 ssch18
757 conn3.22694 disc23.7063 pend.230894 cpu12336.9 u0
r0 10 t7 s3 16 lat-1 d10 cr0 ro22.962 la195 lap.901484 latp0 laav121.387 i741 e0 sz377.236 szs236.316 ti90183.6 th12.1078 io8.21657 ch94.0301 chc4.04791 qd1.00135 ssch7042 conn65.
2886 disc25.6957 pend.234388 cpu1870.75 u0
r0 10 t7 s4 16 lat-1 d10 cr0 ro.654917 la37 lap.891397 latp0 laav28.36 i4208 e0 sz1.60979 szs.986756 ti120089 th.220344 io35.0407 ch27.4328 chc.827944 qd1.00024 ssch4208 conn3.277
6 disc23.4084 pend.231027 cpu1752.13 u0
r1 10 t13 s3 16 lat780 d20 cr1 ro7.24226 la791 lap.9 latp.892 laav497.526 i2505 e0 sz386.345 szs236.917 ti120939 th31.2591 io20.7129 ch92.7267 chc4.38754 qd10.4643 ssch23946 conn6
6.6342 disc22.7589 pend.231219 cpu5746.27 u0
r1 10 t13 s4 16 lat148 d20 cr1 ro1.1355 la147 lap.899684 latp.904714 laav77.6468 i7021 e0 sz1.56844 szs.929839 ti12111.8 th3.55154 io579.682 ch24.8604 chc1.4997 qd46.5746 ssch7057
 conn3.21244 disc20.7869 pend.2297 cpu5322.18 u0
r1 10 t9 s3 16 lat780 d20 cr1 ro7.01629 la781 lap.901395 latp.901395 laav598.453 i1075 e0 sz377.98 szs234.261 ti100782 th15.749 io10.6665 ch93.6306 chc4.08376 qd6.45395 ssch20479
conn65.5071 disc25.3979 pend.231668 cpu4397.09 u0
r1 10 t9 s4 16 lat148 d20 cr1 ro2.32932 la145 lap.900834 latp.909178 laav77.3927 i35619 e0 sz1.58427 szs.976633 ti155662 th1.41608 io228.823 ch27.7296 chc.809779 qd18.3582 ssch712
78 conn3.23342 disc23.6693 pend.2311 cpu49143.7 u0
r5 10 t13 s3 16 lat780 d11 cr5 ro3.90841 la777 lap.900446 latp.910847 laav596.944 i673 e0 sz375.009 szs234.932 ti60836.9 th16.205 io11.0624 ch92.1498 chc5.94573 qd6.74294 ssch6941
 conn63.0905 disc26.0262 pend.232192 cpu1629.57 u0
r5 10 t13 s4 16 lat148 d11 cr5 ro.62268 la144 lap.899541 latp.907798 laav70.4473 i6569 e0 sz1.57695 szs.992693 ti23467 th1.72433 io279.925 ch27.6905 chc.923976 qd20.0094 ssch6588
conn3.22512 disc23.6798 pend.230043 cpu2656.72 u0
r5 10 t9 s3 16 lat780 d11 cr5 ro8.4284 la794 lap.90061 latp.895122 laav576.943 i1640 e55562.7 sz384.684 szs234.527 ti201570 th12.2259 io8.13611 ch74.6627 chc3.42341 qd4.73171 ssch
26145 conn50.4658 disc21.6273 pend.233289 cpu52192.9 u0
r5 10 t9 s4 16 lat148 d11 cr5 ro2.08863 la146 lap.900352 latp.911487 laav89.8798 i7095 e1330.22 sz1.59239 szs.974169 ti100253 th.440214 io70.7709 ch21.8746 chc.762837 qd6.43693 ss
ch28402 conn3.24848 disc17.8499 pend.231876 cpu9288.52 u0
exor5 10 t7 s3 16 lat-1 d11 cr5 ro25.9009 la210 lap.900329 latp0 laav131.116 i913 e1916.7 sz398.783 szs235.563 ti120202 th11.8319 io7.59554 ch92.9173 chc3.8137 qd1.0011 ssch9975 c
onn64.5381 disc25.3423 pend.234545 cpu4452.87 u0
exor5 10 t7 s4 16 lat-1 d11 cr5 ro.897763 la44 lap.896905 latp0 laav30.5335 i3909 e41.319 sz1.6022 szs.99839 ti120050 th.203789 io32.5614 ch25.6209 chc.84496 qd1.00026 ssch5085 co
nn3.27912 disc21.5941 pend.229318 cpu1974.56 u0
exor5 10 t8 s3 16 lat-1 d11 cr5 ro23.9128 la227 lap.90125 latp0 laav149.626 i800 e11678.6 sz390.71 szs241.24 ti120188 th10.1588 io6.65624 ch76.6579 chc3.34966 qd1.00125 ssch12276
conn52.1922 disc21.8409 pend.236825 cpu13452.8 u0
exor5 10 t8 s3 16 lat780 d11 cr5 ro8.70701 la777 lap.900277 latp.907202 laav586.568 i723 e11724.6 sz399.959 szs226.651 ti90488.2 th12.4831 io7.99 ch77.0924 chc5.1264 qd4.75795 ssc
h11296 conn52.4812 disc21.9365 pend.231581 cpu12522 u0
exor5 10 t8 s4 16 lat-1 d11 cr5 ro2.71604 la57 lap.890075 latp0 laav47.8541 i1874 e156.493 sz1.57097 szs.963324 ti90077 th.127669 io20.8044 ch21.9154 chc.887377 qd1.00053 ssch6887
 conn3.21717 disc17.9455 pend.230054 cpu2251.58 u0
exor5 10 t8 s4 16 lat148 d11 cr5 ro1.92481 la146 lap.900188 latp.909123 laav87.4918 i8509 e701.094 sz1.57751 szs.96136 ti111182 th.4716 io76.532 ch22.0649 chc.762969 qd6.86309 ssc
h31346 conn3.21652 disc18.0731 pend.231262 cpu9642.41 u0
exor5n 10 t7 s3 16 lat-1 d11 cr5 ro24.9404 la208 lap.899676 latp0 laav129.292 i927 e0 sz385.518 szs242.051 ti120344 th11.6001 io7.70294 ch90.7976 chc3.78484 qd1.00108 ssch10052 co
nn62.9533 disc24.8151 pend.234544 cpu2823.8 u0
exor5n 10 t7 s4 16 lat-1 d11 cr5 ro.868469 la43 lap.900776 latp0 laav30.2117 i2963 e0 sz1.57138 szs.939919 ti90037.1 th.202 io32.9087 ch25.6326 chc.862732 qd1.00034 ssch3803 conn3
.2379 disc21.6507 pend.232642 cpu1443.34 u0
exor5n 10 t8 s3 16 lat-1 d11 cr5 ro22.3047 la225 lap.900504 latp0 laav151.068 i794 e0 sz388.436 szs241.905 ti120427 th10.004 io6.5932 ch76.598 chc3.4116 qd1.00126 ssch12113 conn52
.1422 disc21.8234 pend.234253 cpu3219.16 u0
exor5n 10 t8 s3 16 lat780 d11 cr5 ro7.90377 la778 lap.90169 latp.907834 laav568.722 i1953 e0 sz400.03 szs245.661 ti241233 th12.6508 io8.09592 ch77.9507 chc3.53921 qd4.63953 ssch29
967 conn53.247 disc21.9967 pend.232426 cpu7045.45 u0
exor5n 10 t8 s4 16 lat-1 d11 cr5 ro2.61631 la58 lap.909413 latp0 laav48.2032 i3102 e0 sz1.56319 szs.915712 ti150179 th.126125 io20.6553 ch21.8317 chc.833387 qd1.00032 ssch11436 co
nn3.20148 disc17.8845 pend.23085 cpu3420.03 u0
exor5n 10 t8 s4 16 lat148 d11 cr5 ro1.8602 la148 lap.901677 latp.90561 laav88.7933 i4837 e0 sz1.58611 szs1.00458 ti60116.3 th.498512 io80.4606 ch21.8594 chc.795872 qd7.32396 ssch1
7935 conn3.24032 disc17.8482 pend.230614 cpu5023.96 u0
r0 10 t13 s1 110 lat-1 d10 cr0 ro10.9142 la67 lap.915101 latp0 laav59.9931 i1331 e0 sz100 szs0 ti80160.6 th6.486 io16.6042 ch40.8418 chc1.2694 qd1.00075 ssch13310 conn16.1564 disc
23.3885 pend.259606 cpu3303.44 u0
r0 10 t13 s1 110 lat-2 d10 cr0 ro3.35535 la100 lap.948958 latp0 laav86.0486 i1391 e0 sz100 szs0 ti60120.7 th9.03781 io23.1368 ch40.5287 chc1.2056 qd2.00288 ssch13912 conn16.1601 d
```

```
isc23.0612 pend.239834 cpu2899.69 u0
r0 10 t13 s1 110 lat-3 d10 cr0 ro3.60804 la149 lap.895263 latp0 laav130.183 i2301 e0 sz100 szs0 ti100194 th8.97085 io22.9654 ch40.4575 chcl.17716 qd3.00391 ssch23021 conn16.1602 d
isc22.9784 pend.239514 cpu4861.19 u0
r0 10 t13 s1 110 lat268 d10 cr0 ro3.77806 la280 lap.901217 latp.894775 laav230.215 i1397 e0 sz100 szs0 ti60271.3 th9.0541 io23.1785 ch40.2444 chcl.63311 qd5.3801 ssch13982 conn16.
1601 disc22.7762 pend.239302 cpu2977.17 u0
r0 10 t13 s2 110 lat-1 d10 cr0 ro.705255 la50 lap.903539 latp0 laav41.6041 i1441 e0 sz10 szs0 ti60131.2 th.936104 io23.9643 ch40.6861 chcl.35061 qd1.00069 ssch1441 conn16.1586 dis
c23.2997 pend.229441 cpu518.672 u0
r0 10 t13 s2 110 lat-10 d10 cr0 ro1.21883 la116 lap.902157 latp0 laav67.1037 i8902 e0 sz10 szs0 ti60098.1 th5.78612 io148.125 ch40.1194 chcl.20541 qd10.0112 ssch8911 conn16.1585 d
isc22.6713 pend.230877 cpu4375.52 u0
r0 10 t13 s2 110 lat-20 d10 cr0 ro1.1498 la204 lap.900425 latp0 laav105.745 i15069 e0 sz10 szs0 ti80312.1 th7.32931 io187.63 ch40.2185 chcl.16765 qd20.0265 ssch15087 conn16.1577 d
isc22.7513 pend.232923 cpu7794.66 u0
r0 10 t13 s2 110 lat-30 d10 cr0 ro1.03933 la305 lap.900229 latp0 laav146.2 i12257 e0 sz10 szs0 ti60302.3 th7.93981 io203.259 ch40.262 chcl.18572 qd30.0734 ssch12286 conn16.1586 di
sc22.7896 pend.233163 cpu6086.58 u0
r0 10 t13 s2 110 lat-40 d10 cr0 ro1.18639 la394 lap.899702 latp0 laav184.899 i21545 e0 sz10 szs0 ti100443 th8.37888 io214.499 ch40.1722 chcl.1554 qd40.0743 ssch21584 conn16.1589 d
isc22.6932 pend.233376 cpu14539.3 u0
r0 10 t13 s2 110 lat-5 d10 cr0 ro.978902 la81 lap.896564 latp0 laav51.8916 i5763 e0 sz10 szs0 ti60083.6 th3.74673 io95.9164 ch40.2876 chcl.24688 qd5.00434 ssch5767 conn16.1592 dis
c22.8562 pend.231119 cpu2253.69 u0
r0 10 t13 s2 110 lat-60 d10 cr0 ro.890886 la572 lap.899924 latp0 laav266.355 i22461 e0 sz10 szs0 ti100662 th8.71617 io223.134 ch40.1438 chcl.16133 qd60.1603 ssch22519 conn16.1589
disc22.6605 pend.233849 cpu11555.7 u0
r0 10 t13 s2 110 lat200 d10 cr0 ro1.17718 la204 lap.899785 latp.895759 laav103.998 i11197 e0 sz10 szs0 ti60212.9 th7.26394 io185.957 ch40.2394 chcl.35219 qd19.7151 ssch11216 connl
6.1595 disc22.7652 pend.233107 cpu6151.8 u0
r0 10 t13 s4 110 lat-10 d10 cr0 ro.990739 la84 lap.898565 latp0 laav47.5822 i20834 e0 sz1.58813 szs.962848 ti100090 th1.2913 io208.153 ch27.6024 chc.690678 qd10.0048 ssch20842 con
n3.24217 disc23.567 pend.230476 cpu12404.1 u0
r0 10 t13 s4 110 lat-20 d10 cr0 ro.935774 la146 lap.899981 latp0 laav73.4698 i16210 e0 sz1.59112 szs.963425 ti60190 th1.67387 io269.314 ch27.7813 chc.666457 qd20.0247 ssch16227 co
nn3.24531 disc23.7377 pend.230624 cpu10099.6 u0
r0 10 t13 s4 110 lat-30 d10 cr0 ro.893814 la213 lap.900333 latp0 laav101.761 i23439 e0 sz1.58701 szs.964498 ti80155.1 th1.81279 io292.421 ch27.8818 chc.650426 qd30.0384 ssch23466
conn3.24082 disc23.8375 pend.230559 cpu14802.1 u0
r0 10 t13 s4 110 lat-40 d10 cr0 ro1.00655 la263 lap.899526 latp0 laav128.214 i24764 e0 sz1.59203 szs.995696 ti80208.8 th1.92004 io308.744 ch27.852 chc.647687 qd40.0646 ssch24804 c
onn3.247 disc23.794 pend.231674 cpu18853.9 u0
r0 10 t13 s4 110 lat-5 d10 cr0 ro.893866 la58 lap.900659 latp0 laav35.7411 i8346 e0 sz1.56578 szs.97687 ti60055.9 th.849989 io138.971 ch27.4735 chc.73302 qd5.003 ssch8351 conn3.20
457 disc23.4976 pend.231461 cpu3576.07 u0
r0 10 t7 s3 16 lat-2 d10 cr0 ro5.21851 la282 lap.89934 latp0 laav198.218 i606 e0 sz392.757 szs239.759 ti60321.8 th15.4128 io10.0461 ch98.2249 chc3.66428 qd2.0066 ssch5748 conn68.2
546 disc26.7472 pend.231215 cpu1278.97 u0
r0 10 t7 s3 16 lat-3 d10 cr0 ro3.99026 la403 lap.899637 latp0 laav290.88 i827 e0 sz398.622 szs240.167 ti80489.1 th15.9989 io10.2747 ch98.0513 chc3.53301 qd3.01088 ssch7863 conn69.
0284 disc25.7663 pend.230963 cpu1721.91 u0
r0 10 t7 s3 16 lat-4 d10 cr0 ro3.23874 la500 lap.900474 latp0 laav378.644 i633 e0 sz386.904 szs229.408 ti60257.8 th15.8764 io10.5049 ch95.861 chc3.88549 qd4.02528 ssch6050 conn66.
89 disc25.9185 pend.231119 cpu1312.16 u0
r0 10 t7 s3 16 lat-5 d10 cr0 ro4.02901 la626 lap.900322 latp0 laav482.684 i1244 e0 sz394.197 szs237.371 ti120560 th15.8888 io10.3185 ch96.8872 chc3.45381 qd5.0201 ssch11910 conn67
.7891 disc25.8691 pend.231518 cpu2603.27 u0
r0 10 t7 s3 16 lat-1 d10 cr0 ro22.962 la195 lap.901484 latp0 laav121.387 i741 e0 sz377.236 szs236.316 ti90183.6 th12.1078 io8.21657 ch94.0301 chc4.04791 qd1.00135 ssch7042 conn65.
2886 disc25.6957 pend.234388 cpu1870.75 u0
r0 10 t7 s4 16 lat-10 d10 cr0 ro.88872 la87 lap.90126 latp0 laav48.3705 i12310 e0 sz1.58741 szs.966517 ti60160 th1.26882 io204.621 ch27.6225 chc.695207 qd10.0081 ssch12317 conn3.2
4203 disc23.5997 pend.229292 cpu7065.41 u0
r0 10 t7 s4 16 lat-20 d10 cr0 ro.895941 la163 lap.899593 latp0 laav78.081 i15264 e0 sz1.58045 szs.965373 ti60142.2 th1.56686 io253.798 ch27.7319 chc.673439 qd20.0262 ssch15284 con
n3.23 disc23.7116 pend.229402 cpu9353.84 u0
r0 10 t7 s4 16 lat-30 d10 cr0 ro.768451 la250 lap.900178 latp0 laav109.594 i16336 e0 sz1.57952 szs.966781 ti60238.4 th1.67324 io271.189 ch27.7002 chc.67404 qd30.0551 ssch16364 con
n3.22896 disc23.6815 pend.230125 cpu8942.36 u0
r0 10 t7 s4 16 lat-40 d10 cr0 ro.908068 la344 lap.900106 latp0 laav140.287 i34033 e0 sz1.58311 szs.960434 ti120661 th1.74424 io282.056 ch27.9113 chc.66123 qd40.047 ssch34070 conn3
.23476 disc23.8807 pend.230269 cpu22798.3 u0
r0 10 t7 s4 16 lat-5 d10 cr0 ro.98066 la59 lap.897506 latp0 laav36.408 i10908 e0 sz1.59571 szs.986296 ti80112.2 th.848712 io136.159 ch27.4932 chc.732292 qd5.00229 ssch10911 conn3.
25407 disc23.4663 pend.230426 cpu5515.88 u0
r1 10 t13 s1 110 lat-1 d20 cr1 ro15.2739 la66 lap.901883 latp0 laav59.2635 i1009 e0 sz100 szs0 ti60062.7 th6.56215 io16.7991 ch37.5749 chcl.5554 qd1.00099 ssch10090 conn16.1579 di
sc20.1441 pend.231364 cpu3072.1 u0
r1 10 t13 s1 110 lat-2 d20 cr1 ro12.154 la67 lap.907957 latp0 laav58.4596 i2727 e0 sz100 szs0 ti80101.4 th13.2986 io34.0443 ch40.6704 chcl.34514 qd2.00147 ssch27280 conn16.1573 di
sc23.155 pend.24304 cpu7249.84 u0
r1 10 t13 s1 110 lat-3 d20 cr1 ro7.21598 la89 lap.904019 latp0 laav69.9565 i2563 e0 sz100 szs0 ti60120.4 th16.6528 io42.6311 ch38.395 chcl.26894 qd3.00351 ssch25640 conn16.1581 di
sc20.8425 pend.2369 cpu6070.17 u0
r1 10 t13 s1 110 lat-4 d20 cr1 ro5.428 la113 lap.897354 latp0 laav84.4209 i2835 e0 sz100 szs0 ti60128 th18.4177 io47.1494 ch36.9357 chcl.23684 qd4.00564 ssch28369 conn16.1599 disc
19.3927 pend.235579 cpu6488.48 u0
r1 10 t13 s1 110 lat-6 d20 cr1 ro4.5305 la166 lap.903161 latp0 laav119.975 i3986 e0 sz100 szs0 ti80160.2 th19.424 io49.7254 ch36.347 chcl.22342 qd6.00903 ssch39907 conn16.1584 dis
c18.7665 pend.234889 cpu9159.99 u0
r1 10 t13 s1 110 lat268 d20 cr1 ro3.81745 la267 lap.899065 latp.903902 laav184.82 i3101 e0 sz100 szs0 ti60150.3 th20.1384 io51.5542 ch35.7852 chcl.45479 qd9.61754 ssch31064 conn16
.1607 disc18.2555 pend.231738 cpu6997.21 u0
r1 10 t13 s2 110 lat-1 d20 cr1 ro1.04057 la46 lap.887811 latp0 laav38.7214 i2576 e0 sz10 szs0 ti100086 th1.00538 io25.7378 ch37.3707 chcl.55027 qd1.00039 ssch2576 conn16.159 disc1
9.9779 pend.228571 cpu1069.98 u0
```

```
rl 10 t13 s2 110 lat-10 d20 crl rol.63572 la64 lap.897562 latp0 laav46.6097 i17025 e0 sz10 szs0 ti80094.4 th8.30319 io212.562 ch38.8742 chcl.39662 qd10.0059 ssch17032 conn16.158 d
isc21.3616 pend.231403 cpu10391.4 u0
rl 10 t13 s2 110 lat-120 d20 crl ro.998273 la491 lap.900194 latp0 laav230.737 i31016 e0 sz10 szs0 ti60895.7 th19.8957 io509.33 ch35.5888 chcl.26179 qd120.464 ssch31132 conn16.1594
 discl7.9866 pend.231993 cpu17574.1 u0
rl 10 t13 s2 110 lat-20 d20 crl rol.29106 la97 lap.901098 latp0 laav58.5605 i8483 e0 sz10 szs0 ti25109.7 th13.1968 io337.837 ch37.941 chcl.36602 qd20.0472 ssch8498 conn16.159 disc
20.4072 pend.231885 cpu4297.37 u0
rl 10 t13 s2 110 lat-40 d20 crl rol.17599 la168 lap.900102 latp0 laav89.5311 i8873 e0 sz10 szs0 ti20136.9 th17.2123 io440.634 ch36.9337 chcl.34399 qd40.1803 ssch8906 conn16.157 di
sc19.3253 pend.230963 cpu4958.82 u0
rl 10 t13 s2 110 lat-60 d20 crl rol.04756 la247 lap.900317 latp0 laav122.614 i12108 e0 sz10 szs0 ti25371.7 th18.6416 io477.224 ch36.2127 chcl.31462 qd60.2973 ssch12159 conn16.1606
 disc18.6516 pend.232482 cpu6295.86 u0
rl 10 t13 s2 110 lat-80 d20 crl rol.02431 la332 lap.899883 latp0 laav158.468 i30099 e0 sz10 szs0 ti60298.6 th19.4987 io499.166 ch35.9097 chcl.25027 qd80.2126 ssch30169 conn16.1583
 disc18.2853 pend.233526 cpu16479.4 u0
rl 10 t13 s2 110 lat200 d20 crl rol.48289 la202 lap.899572 latp.898323 laav108.062 i5704 e0 sz10 szs0 ti12545.1 th17.761 io454.681 ch36.2732 chc3.25273 qd51.9111 ssch5742 conn16.1
59 disc18.705 pend.231211 cpu4802 u0
rl 10 t7 s3 16 lat-1 d20 crl ro28.3821 la203 lap.899859 latp0 laav126.968 i1709 e0 sz392.992 szs236.015 ti90282.1 th12.0556 io7.85316 ch93.631 chc4.44212 qd1.00141 ssch7335 conn67.
0923 disc23.4308 pend.230924 cpu2302.04 u0
rl 10 t7 s3 16 lat-10 d20 crl ro7.41292 la762 lap.900135 latp0 laav539.188 i1483 e0 sz397.636 szs238.223 ti80607.6 th28.5766 io18.3978 ch95.7659 chc3.67781 qd10.0674 ssch15652 con
n68.8772 disc23.6756 pend.230665 cpu3672.81 u0
rl 10 t7 s3 16 lat-2 d20 crl ro24.3435 la216 lap.901739 latp0 laav138.982 i1150 e0 sz395.217 szs233.114 ti80200.7 th22.1368 io14.339 ch96.9851 chc3.40439 qd2.00348 ssch12132 conn6
7.9881 disc25.7417 pend.232018 cpu3314.31 u0
rl 10 t7 s3 16 lat-3 d20 crl ro15.3388 la294 lap.9 latp0 laav187.484 i1600 e0 sz398.989 szs230.14 ti100487 th24.8159 io15.9224 ch96.8884 chc3.27476 qd3.00562 ssch17206 conn68.4666
 disc25.0257 pend.231971 cpu4275.09 u0
rl 10 t7 s3 16 lat-4 d20 crl ro12.5302 la355 lap.900713 latp0 laav227.831 i2105 e0 sz383.348 szs240.341 ti120436 th26.1727 io17.4781 ch94.4707 chc3.19076 qd4.0076 ssch21866 conn66
.6769 disc24.5411 pend.232438 cpu5286.51 u0
rl 10 t7 s3 16 lat-6 d20 crl ro9.05292 la497 lap.900407 latp0 laav324.507 i1477 e0 sz391.414 szs240.545 ti80516.3 th28.0474 io18.3441 ch94.6888 chc3.45777 qd6.02437 ssch15293 conn
67.6014 disc23.7628 pend.231066 cpu3600.67 u0
rl 10 t7 s4 16 lat-10 d20 crl rol.2904 la55 lap.901971 latp0 laav34.5171 i11466 e0 sz1.57841 szs.979558 ti40075.2 th1.76407 io286.112 ch26.26 chc.896487 qd10.0087 ssch12626 conn3.
22773 disc22.2075 pend.230078 cpu7178.12 u0
rl 10 t7 s4 16 lat-20 d20 crl rol.20264 la86 lap.899735 latp0 laav47.5956 i16609 e0 sz1.57969 szs.958429 ti40047.7 th2.55916 io414.731 ch26.3216 chc.828242 qd20.0241 ssch18256 con
n3.22763 disc22.255 pend.230282 cpu11742.1 u0
rl 10 t7 s4 16 lat-40 d20 crl ro.964686 la158 lap.899722 latp0 laav75.8338 i31361 e0 sz1.58267 szs.97109 ti60159.3 th3.22282 io521.299 ch25.9167 chc.78415 qd40.051 ssch34534 conn3
.23292 disc21.8344 pend.230329 cpu19140.7 u0
rl 10 t7 s4 16 lat-60 d20 crl ro.958985 la239 lap.900165 latp0 laav107.102 i55716 e0 sz1.59236 szs.981172 ti100286 th3.45573 io555.569 ch25.7732 chc.780143 qd60.0646 ssch61315 con
n3.24677 disc21.6683 pend.230828 cpu34985.9 u0
rl 10 t7 s4 16 lat-80 d20 crl ro.924845 la324 lap.899718 latp0 laav137.769 i34593 e0 sz1.57234 szs.939613 ti60744.7 th3.49773 io569.481 ch25.6817 chc.803737 qd80.185 ssch38138 con
n3.21715 disc21.6027 pend.231409 cpu21306.3 u0
rl 10 t7 s4 16 lat-1 d20 crl rol.1807 la35 lap.901693 latp0 laav26.5444 i3367 e0 sz1.56638 szs.951905 ti90045.5 th.22879 io37.3922 ch24.8815 chcl.08594 qd1.0003 ssch3705 conn3.206
11 disc20.9323 pend.230055 cpu1722.05 u0
rl 10 t8 s3 16 lat-2 d20 crl ro12.7413 la278 lap.899876 latp0 laav197.837 i809 e0 sz395.726 szs230.404 ti80380.2 th15.558 io10.0647 ch97.0105 chc2.97011 qd2.00494 ssch14786 conn67
.8461 disc26.058 pend.230731 cpu3291.64 u0
rl 10 t8 s3 16 lat-3 d20 crl ro9.41733 la382 lap.902468 latp0 laav282.131 i851 e0 sz393.031 szs233.047 ti80421.3 th16.2459 io10.5818 ch97.119 chc3.03891 qd3.01058 ssch15346 conn67
.9386 disc26.2374 pend.231236 cpu3359.14 u0
rl 10 t8 s3 16 lat-5 d20 crl ro7.78226 la589 lap.899851 latp0 laav448.055 i669 e0 sz391.049 szs223.265 ti60452.3 th16.9046 io11.0666 ch94.8819 chc3.43829 qd5.03737 ssch12035 conn6
7.1324 disc24.9378 pend.230315 cpu2616.18 u0
rl 10 t8 s3 16 lat-1 d20 crl ro41.1312 la223 lap.900187 latp0 laav139.854 i1072 e0 sz387.08 szs233.077 ti150404 th10.777 io7.12746 ch95.2532 chc3.34393 qd1.00093 ssch19436 conn66.
674 disc25.8446 pend.230816 cpu5496.81 u0
rl 10 t8 s4 16 lat-10 d20 crl rol.96273 la85 lap.899255 latp0 laav50.8573 i11675 e0 sz1.59143 szs.976957 ti60128.6 th1.20705 io194.167 ch27.7042 chc.840656 qd10.0086 ssch22193 con
n3.24216 disc23.6588 pend.230225 cpu10772.8 u0
rl 10 t8 s4 16 lat-20 d20 crl rol.81778 la150 lap.899976 latp0 laav77.7727 i25451 e0 sz1.58135 szs.969057 ti100168 th1.56952 io254.084 ch27.8575 chc.777786 qd20.0157 ssch48433 con
n3.22708 disc23.8177 pend.230053 cpu26050.5 u0
rl 10 t8 s4 16 lat-30 d20 crl rol.51382 la217 lap.899682 latp0 laav105.338 i16982 e0 sz1.56695 szs.958673 ti60165.8 th1.72765 io282.254 ch27.7756 chc.78234 qd30.053 ssch32302 conn
3.20754 disc23.7539 pend.230465 cpu14630.9 u0
rl 10 t8 s4 16 lat-40 d20 crl rol.40575 la315 lap.900043 latp0 laav137.832 i34644 e0 sz1.58674 szs.966957 ti120367 th1.78397 io287.821 ch27.8915 chc.780023 qd40.0462 ssch65857 con
n3.23728 disc23.8411 pend.230828 cpu27303 u0
rl 10 t8 s4 16 lat-5 d20 crl ro2.13219 la61 lap.902366 latp0 laav39.5674 i10017 e0 sz1.58481 szs.969651 ti80122.4 th.773962 io125.021 ch27.5587 chc.90235 qd5.0025 ssch19039 conn3.
23515 disc23.5418 pend.229101 cpu8392.07 u0
rl 10 t8 s4 16 lat-1 d20 crl ro2.15065 la41 lap.898073 latp0 laav33.1545 i2698 e0 sz1.59081 szs.949865 ti90030.6 th.186221 io29.9676 ch27.2992 chcl.08591 qd1.00037 ssch5129 conn3.
24797 disc23.3078 pend.229522 cpu2048.25 u0
rl 10 t9 s1 110 lat-1 d20 crl ro23.7551 la79 lap.90024 latp0 laav71.8749 i832 e0 sz100 szs0 ti60115.8 th5.40624 io13.84 ch40.6527 chcl.33362 qd1.0012 ssch16640 conn16.148 disc23.3
471 pend.230969 cpu4497.08 u0
rl 10 t9 s1 110 lat-2 d20 crl ro6.65526 la100 lap.935391 latp0 laav85.753 i1393 e0 sz100 szs0 ti60108.1 th9.05269 io23.1749 ch40.3174 chcl.11262 qd2.00287 ssch27880 conn16.1455 di
sc22.9439 pend.233494 cpu5806.27 u0
rl 10 t9 s1 110 lat-3 d20 crl ro6.36716 la148 lap.845661 latp0 laav127.49 i1406 e0 sz100 szs0 ti60100.5 th9.13834 io23.3942 ch40.1976 chcl.13216 qd3.0064 ssch28141 conn16.1449 dis
c22.8148 pend.234767 cpu5812.09 u0
rl 10 t9 s1 110 lat268 d20 crl ro6.97658 la279 lap.900648 latp.898488 laav212.635 i1389 e0 sz100 szs0 ti60160.2 th9.0189 io23.0884 ch40.2343 chcl.50164 qd4.95104 ssch27813 conn16.
```

```
1451 disc22.861 pend.233118 cpu5824.14 u0
r1 10 t9 s2 110 lat-1 d20 cr1 ro2.51685 la54 lap.892479 latp0 laav46.9336 i1702 e0 sz10 szs0 ti80133.9 th.829666 io21.2394 ch40.4852 chc1.44619 qd1.00059 ssch3404 conn16.1492 disc
23.2104 pend.229716 cpu1226.12 u0
r1 10 t9 s2 110 lat-10 d20 cr1 ro2.38107 la120 lap.900546 latp0 laav71.0267 i11184 e0 sz10 szs0 ti80098.2 th5.45424 io139.629 ch40.3331 chc1.21833 qd10.0089 ssch22378 conn16.1458
disc22.9735 pend.230924 cpu9855.31 u0
r1 10 t9 s2 110 lat-20 d20 cr1 ro2.1642 la210 lap.900034 latp0 laav109.592 i14507 e0 sz10 szs0 ti80335.1 th7.05395 io180.581 ch40.3846 chc1.16934 qd20.0276 ssch29048 conn16.1462 d
isc22.999 pend.231918 cpu14141.7 u0
r1 10 t9 s2 110 lat-30 d20 cr1 ro2.01732 la295 lap.89974 latp0 laav146.947 i16222 e0 sz10 szs0 ti80330.9 th7.88827 io201.94 ch40.2482 chc1.14463 qd30.0555 ssch32495 conn16.1467 di
sc22.8581 pend.231794 cpu16035.6 u0
r1 10 t9 s2 110 lat-40 d20 cr1 ro1.63796 la389 lap.899739 latp0 laav188.867 i21100 e0 sz10 szs0 ti100569 th8.19558 io209.807 ch40.3267 chc1.12995 qd40.0758 ssch42264 conn16.1452 d
isc22.9405 pend.231517 cpu16671.1 u0
r1 10 t9 s2 110 lat-5 d20 cr1 ro2.61177 la83 lap.903196 latp0 laav55.5331 i5382 e0 sz10 szs0 ti60106.4 th3.4977 io89.5412 ch40.2843 chc1.31357 qd5.00465 ssch10767 conn16.1456 disc
22.9519 pend.230452 cpu3939.94 u0
r1 10 t9 s2 110 lat-60 d20 cr1 ro1.39577 la580 lap.900068 latp0 laav270.79 i13253 e0 sz10 szs0 ti61270.4 th8.44935 io216.303 ch40.3588 chc1.17132 qd60.2716 ssch26609 conn16.1472 d
isc22.9657 pend.232227 cpu9134.22 u0
r1 10 t9 s2 110 lat200 d20 cr1 ro2.24985 la197 lap.900458 latp.906038 laav105.451 i18116 e0 sz10 szs0 ti100186 th7.0634 io180.823 ch40.2597 chc1.23105 qd19.2829 ssch36257 conn16.1
459 disc22.8907 pend.23091 cpu18973 u0
r1 10 t9 s4 110 lat-1 d20 cr1 ro2.29313 la42 lap.898698 latp0 laav33.7869 i1767 e0 sz1.59253 szs.971181 ti60049.2 th.183053 io29.4259 ch27.4406 chc1.03223 qd1.00057 ssch3534 conn3
.24759 disc23.4602 pend.228908 cpu1282.2 u0
r1 10 t9 s4 110 lat-10 d20 cr1 ro1.89674 la84 lap.897723 latp0 laav50.5351 i11773 e0 sz1.58201 szs.985328 ti60080.8 th1.21093 io195.953 ch27.6324 chc.825088 qd10.0085 ssch23564 co
nn3.22888 disc23.5975 pend.230757 cpu9981.45 u0
r1 10 t9 s4 110 lat-20 d20 cr1 ro1.82327 la146 lap.900253 latp0 laav76.9875 i15440 e0 sz1.57494 szs.947293 ti60143.1 th1.57937 io256.721 ch27.7337 chc.769654 qd20.0259 ssch30911 c
onn3.21774 disc23.6943 pend.230856 cpu15475.1 u0
r1 10 t9 s4 110 lat-30 d20 cr1 ro1.32491 la202 lap.899463 latp0 laav104.391 i4306 e0 sz1.57524 szs.951948 ti15430.7 th1.7171 io279.053 ch27.9022 chc.845771 qd30.209 ssch8655 conn3
.2176 disc23.8692 pend.231243 cpu3109.07 u0
r1 10 t9 s4 110 lat-40 d20 cr1 ro1.32312 la286 lap.900056 latp0 laav133.722 i17858 e0 sz1.58159 szs.953024 ti60418.5 th1.82607 io295.572 ch27.9288 chc.748602 qd40.0896 ssch35769 c
onn3.22827 disc23.8876 pend.231555 cpu13395.5 u0
r1 10 t9 s4 110 lat-5 d20 cr1 ro2.21947 la61 lap.902265 latp0 laav39.4625 i7551 e0 sz1.58496 szs.983323 ti60100.9 th.777859 io125.639 ch27.5223 chc.904422 qd5.00331 ssch15106 conn
3.23487 disc23.4948 pend.230368 cpu5872.32 u0
r5 10 t13 s1 110 lat-1 d11 cr5 ro11.3352 la67 lap.905592 latp0 laav59.9421 i1663 e0 sz100 szs0 ti100150 th6.48636 io16.6051 ch40.5656 chc1.27696 qd1.0006 ssch16630 conn16.1577 dis
c23.1211 pend.243831 cpu4351.58 u0
r5 10 t13 s1 110 lat-2 d11 cr5 ro4.50801 la99 lap.826176 latp0 laav81.5456 i1467 e0 sz100 szs0 ti60116.5 th9.53227 io24.4026 ch40.2647 chc1.21179 qd2.00273 ssch14678 conn16.1616 d
isc22.82 pend.234687 cpu3320.55 u0
r5 10 t13 s1 110 lat-4 d11 cr5 ro4.15937 la188 lap.899866 latp0 laav160.013 i1498 e0 sz100 szs0 ti60190.3 th9.72177 io24.8877 ch40.0174 chc1.25828 qd4.01068 ssch14987 conn16.1572
disc22.5523 pend.235032 cpu3334.59 u0
r5 10 t13 s1 110 lat268 d11 cr5 ro2.94511 la269 lap.900048 latp.900048 laav224.715 i2071 e0 sz100 szs0 ti80273.1 th10.0779 io25.7994 ch39.7233 chc1.43783 qd5.85128 ssch20747 conn1
6.1602 disc22.2857 pend.233987 cpu4499.94 u0
r5 10 t13 s2 110 lat-1 d11 cr5 ro.761049 la50 lap.890287 latp0 laav41.8532 i1431 e0 sz10 szs0 ti60079.9 th.930402 io23.8183 ch40.9169 chc1.37684 qd1.0007 ssch1431 conn16.1538 disc
23.5301 pend.235606 cpu542.523 u0
r5 10 t13 s2 110 lat-11 d11 cr5 ro1.03698 la117 lap.898769 latp0 laav67.92 i12928 e0 sz10 szs0 ti80120.9 th6.30297 io161.356 ch40.2002 chc1.21006 qd11.0094 ssch12940 conn16.1581 d
isc22.7322 pend.230212 cpu4563.46 u0
r5 10 t13 s2 110 lat-22 d11 cr5 ro.919469 la200 lap.899603 latp0 laav104.356 i12614 e0 sz10 szs0 ti60174 th8.1885 io209.625 ch40.2157 chc1.18454 qd22.0384 ssch12634 conn16.1605 di
sc22.7555 pend.231229 cpu4364.6 u0
r5 10 t13 s2 110 lat-33 d11 cr5 ro.822303 la294 lap.90023 latp0 laav144.318 i27388 e0 sz10 szs0 ti120344 th8.88986 io227.58 ch40.1801 chc1.14896 qd33.0398 ssch27420 conn16.1602 di
sc22.7203 pend.231068 cpu9346.55 u0
r5 10 t13 s2 110 lat-44 d11 cr5 ro.767645 la389 lap.900325 latp0 laav185.579 i14189 e0 sz10 szs0 ti60272.7 th9.19584 io235.414 ch40.2438 chc1.18828 qd44.1364 ssch14232 conn16.1603
 disc22.7787 pend.233102 cpu4822.15 u0
r5 10 t13 s2 110 lat-5 d11 cr5 ro.841438 la80 lap.897993 latp0 laav51.0889 i9764 e0 sz10 szs0 ti100072 th3.81133 io97.57 ch40.3146 chc1.24967 qd5.00256 ssch9765 conn16.1584 disc22
.8712 pend.232445 cpu3517.8 u0
r5 10 t13 s2 110 lat-66 d11 cr5 ro.640518 la567 lap.900071 latp0 laav265.378 i19765 e0 sz10 szs0 ti80801.5 th9.55515 io244.612 ch40.232 chc1.17719 qd66.2204 ssch19832 conn16.1548
disc22.7583 pend.232565 cpu6569.03 u0
r5 10 t13 s2 110 lat200 d11 cr5 ro.93057 la197 lap.900494 latp.90679 laav99.4472 i16223 e0 sz10 szs0 ti80316.6 th7.89016 io201.988 ch40.2069 chc1.31036 qd20.2483 ssch16243 conn16.
1591 disc22.7534 pend.231823 cpu5650.72 u0
r5 10 t7 s3 16 lat-2 d11 cr5 ro7.7907 la283 lap.899408 latp0 laav196.935 i1014 e3235.93 sz390.566 szs229.73 ti100418 th15.4057 io10.0978 ch91.6424 chc3.21165 qd2.00394 ssch11083 c
onn63.1781 disc25.4343 pend.230707 cpu5470.34 u0
r5 10 t7 s3 16 lat-4 d11 cr5 ro4.21931 la525 lap.897764 latp0 laav383.179 i626 e2369.68 sz398.297 szs253.855 ti60459.2 th16.1094 io10.3541 ch93.4352 chc3.8287 qd4.02556 ssch6814 c
onn64.9666 disc25.3834 pend.230791 cpu3608.66 u0
r5 10 t7 s3 16 lat-6 d11 cr5 ro4.22572 la725 lap.900592 latp0 laav567.386 i845 e2774.39 sz390.217 szs237.551 ti80592 th15.982 io10.4849 ch90.7573 chc3.67889 qd6.0426 ssch9293 conn
63.2664 disc24.742 pend.232089 cpu4501.05 u0
r5 10 t7 s3 16 lat-1 d11 cr5 ro26.5285 la208 lap.900439 latp0 laav131.445 i683 e2249.52 sz398.141 szs233.776 ti90151.8 th11.7826 io7.57611 ch91.3708 chc4.11648 qd1.00146 ssch7594
conn63.2539 disc25.1296 pend.233026 cpu4109.86 u0
r5 10 t7 s4 16 lat-11 d11 cr5 ro.817174 la106 lap.900854 latp0 laav56.3015 i15576 e297.186 sz1.58109 szs.957883 ti80129.5 th1.20055 io194.385 ch25.8626 chc.698252 qd11.0078 ssch20
248 conn3.23577 disc21.8449 pend.230998 cpu7756.29 u0
r5 10 t7 s4 16 lat-22 d11 cr5 ro.743794 la199 lap.900806 latp0 laav93.1213 i23582 e459.579 sz1.59092 szs.983939 ti100316 th1.4609 io235.078 ch25.937 chc.672224 qd22.0205 ssch30704
 conn3.24732 disc21.9042 pend.230374 cpu11605.2 u0
```

```
r5 10 t7 s4 16 lat-33 dll cr5 ro.681618 la289 lap.900211 latp0 laav129.944 i15187 e298.306 sz1.59176 szs.971609 ti60519.8 th1.56031 io250.943 ch25.978 chc.692001 qd33.0717 ssch198
29 conn3.239 disc21.9463 pend.231993 cpu7440.97 u0
r5 10 t7 s4 16 lat-44 dll cr5 ro.642475 la376 lap.899948 latp0 laav164.73 i21299 e421.526 sz1.58176 szs.965832 ti80794.5 th1.62884 io263.619 ch25.8493 chc.686362 qd44.0909 ssch277
34 conn3.23948 disc21.8129 pend.232513 cpu10350.1 u0
r5 10 t7 s4 16 lat-5 dll cr5 ro.961517 la67 lap.900159 latp0 laav39.6424 i7532 e127.651 sz1.58192 szs.980185 ti60068.7 th.774829 io125.39 ch25.7289 chc.747658 qd5.00332 ssch9641 c
onn3.21442 disc21.7467 pend.231385 cpu3809.34 u0
r5 10 t7 s4 16 lat-1 dll cr5 ro.924431 la44 lap.900286 latp0 laav30.4907 i4894 e93.574 sz1.59338 szs.961543 ti150075 th.202972 io32.6104 ch25.5465 chc.832928 qd1.0002 ssch6427 con
n3.24298 disc21.5582 pend.228596 cpu2543.29 u0
r5 10 t8 s3 16 lat-2 dll cr5 ro10.2957 la348 lap.900901 latp0 laav252.23 i1110 e35558.2 sz407.169 szs246.236 ti140664 th12.5509 io7.89116 ch78.1838 chc2.73906 qd2.0036 ssch17266 c
onn53.5191 disc21.9591 pend.232537 cpu33880 u0
r5 10 t8 s3 16 lat-3 dll cr5 ro8.2237 la475 lap.9 latp0 laav358.376 i670 e20614.4 sz388.809 szs239.849 ti80510.6 th12.6392 io8.32189 ch76.4364 chc3.08371 qd3.01343 ssch10255 conn5
2.1363 disc21.6689 pend.233583 cpu19509.4 u0
r5 10 t8 s3 16 lat-5 dll cr5 ro8.22198 la745 lap.896047 latp0 laav586.453 i683 e20225.7 sz376.6 szs234.925 ti80671.3 th12.455 io8.46645 ch75.7652 chc3.2365 qd5.0366 ssch10377 conn
50.9275 disc22.2308 pend.232897 cpu19389.7 u0
r5 10 t8 s3 16 lat-1 dll cr5 ro23.7742 la220 lap.898914 latp0 laav150.17 i1197 e35558.3 sz392.126 szs240.523 ti180488 th10.1586 io6.63203 ch76.6334 chc3.07127 qd1.00084 ssch18404
conn52.2654 disc21.7497 pend.235177 cpu35855.6 u0
r5 10 t8 s4 16 lat-10 dll cr5 ro1.73291 la185 lap.900078 latp0 laav110.674 i9012 e1532.18 sz1.58799 szs.978986 ti100166 th.558097 io89.9706 ch22.0802 chc.699433 qd10.0111 ssch3328
8 conn3.24279 disc18.0577 pend.231575 cpu10805.1 u0
r5 10 t8 s4 16 lat-20 dll cr5 ro1.46901 la352 lap.8996 latp0 laav188.123 i12744 e2204.97 sz1.5747 szs.938425 ti120295 th.651653 io105.94 ch22.1589 chc.685329 qd20.0314 ssch47231 c
onn3.21773 disc18.15 pend.23149 cpu15024.4 u0
r5 10 t8 s4 16 lat-40 dll cr5 ro1.34957 la745 lap.900062 latp0 laav349.655 i11376 e2033.26 sz1.58448 szs.954154 ti100633 th.699674 io113.045 ch22.1529 chc.723549 qd40.1406 ssch421
36 conn3.2477 disc18.1155 pend.231345 cpu13303.9 u0
r5 10 t8 s4 16 lat-5 dll cr5 ro2.01709 la116 lap.899951 latp0 laav73.4483 i4069 e660.617 sz1.56992 szs.943912 ti60059.4 th.415474 io67.7496 ch22.0242 chc.740487 qd5.00614 ssch1498
0 conn3.20749 disc18.0472 pend.231947 cpu4970.32 u0
r5 10 t8 s4 16 lat-1 dll cr5 ro2.6757 la57 lap.891445 latp0 laav48.4377 i3086 e503.625 sz1.59559 szs.94181 ti150143 th.128107 io20.5537 ch21.8946 chc.841877 qd1.00032 ssch11450 co
nn3.25162 disc17.8882 pend.231149 cpu3984.48 u0
r5 10 t9 s1 110 lat-1 dll cr5 ro12.44 la67 lap.910295 latp0 laav60.0034 i1661 e9703.57 sz100 szs0 ti100180 th6.47662 io16.5802 ch40.2824 chc1.27141 qd1.0006 ssch18271 conn16.1581
disc22.8321 pend.257247 cpu14146.9 u0
r5 10 t9 s1 110 lat-2 dll cr5 ro3.5846 la100 lap.946159 latp0 laav85.8222 i1393 e9912.85 sz100 szs0 ti60108.1 th9.0527 io23.1749 ch40.587 chc1.2164 qd2.00287 ssch15334 conn16.1585
 disc23.0979 pend.236918 cpu11373.1 u0
r5 10 t9 s1 110 lat-3 dll cr5 ro4.11784 la149 lap.952722 latp0 laav128.673 i1396 e9791.93 sz100 szs0 ti60185.1 th9.06059 io23.1951 ch40.1174 chc1.25091 qd3.00645 ssch15371 conn16.
1584 disc22.6204 pend.240586 cpu11426.5 u0
r5 10 t9 s1 110 lat268 dll cr5 ro4.06988 la266 lap.903483 latp.904935 laav230.28 i1378 e9768.3 sz100 szs0 ti60223.1 th8.93812 io22.8816 ch40.3068 chc1.77467 qd5.31277 ssch15173 co
nn16.1593 disc22.8234 pend.235602 cpu11298.4 u0
r5 10 t9 s2 110 lat-1 dll cr5 ro3.69701 la89 lap.897959 latp0 laav81.5787 i735 e993.36 sz10 szs0 ti60105 th.477679 io12.2286 ch36.6223 chc1.44301 qd1.00136 ssch2940 conn16.1567 di
sc19.2091 pend.229094 cpu1859.14 u0
r5 10 t9 s2 110 lat-10 dll cr5 ro2.37653 la314 lap.899651 latp0 laav190.012 i3153 e4581.52 sz10 szs0 ti60243.5 th2.04444 io52.3376 ch36.4097 chc1.30682 qd10.0317 ssch12637 conn16.
159 disc18.9462 pend.232004 cpu7595.85 u0
r5 10 t9 s2 110 lat-132 dll cr5 ro1.15346 la2000 lap1 latp0 laav1812.09 i5695 e10349.6 sz10 szs0 ti84307.5 th2.63869 io67.5503 ch36.3819 chc1.61604 qd135.06 ssch23115 conn16.1597
disc18.9035 pend.232216 cpu13596.4 u0
r5 10 t9 s2 110 lat-20 dll cr5 ro1.99956 la609 lap.900224 latp0 laav327.594 i8525 e12949.3 sz10 szs0 ti140696 th2.36687 io60.5918 ch36.4816 chc1.22886 qd20.0469 ssch34151 conn16.1
594 disc19.0018 pend.234925 cpu20381.9 u0
r5 10 t9 s2 110 lat-40 dll cr5 ro1.61506 la1188 lap.900151 latp0 laav596.622 i5333 e8738.11 sz10 szs0 ti81172.1 th2.5664 io65.7 ch36.3179 chc1.34092 qd40.3 ssch21432 conn16.1585 d
isc18.842 pend.233299 cpu12679.6 u0
r5 10 t9 s2 110 lat-5 dll cr5 ro2.78434 la204 lap.901319 latp0 laav127.437 i2351 e3341.33 sz10 szs0 ti60135 th1.52716 io39.0954 ch36.4993 chc1.32088 qd5.01063 ssch9409 conn16.1581
 disc19.0562 pend.230547 cpu5728.97 u0
r5 10 t9 s2 110 lat200 dll cr5 ro3.33065 la200 lap.900128 latp.901413 laav126.808 i3114 e4401.69 sz10 szs0 ti80141.2 th1.51783 io38.8564 ch36.36 chc1.48639 qd4.97431 ssch12466 con
n16.1587 disc18.9102 pend.229981 cpu7625.8 u0
r5 10 t9 s4 110 lat-1 dll cr5 ro2.87246 la58 lap.89181 latp0 laav50.4289 i1978 e350.202 sz1.55056 szs.901477 ti100113 th.119669 io19.7576 ch21.6847 chc.804393 qd1.00051 ssch7912 c
onn3.1851 disc17.7458 pend.22997 cpu2616.01 u0
r5 10 t9 s4 110 lat-10 dll cr5 ro1.81561 la195 lap.901278 latp0 laav115.828 i6894 e1328.16 sz1.58515 szs.944463 ti80189.6 th.532332 io85.9713 ch21.9578 chc.707047 qd10.0145 ssch27
603 conn3.23708 disc17.9391 pend.231377 cpu8495.24 u0
r5 10 t9 s4 110 lat-20 dll cr5 ro1.50226 la342 lap.899804 latp0 laav196.464 i6111 e1155.28 sz1.56292 szs.913503 ti60558.8 th.616072 io100.91 ch21.8945 chc.712376 qd20.0655 ssch244
98 conn3.2015 disc17.9023 pend.23172 cpu7344.58 u0
r5 10 t9 s4 110 lat-5 dll cr5 ro2.18912 la122 lap.899562 latp0 laav78.0316 i6394 e1169.58 sz1.56631 szs.965011 ti100189 th.390474 io63.8196 ch21.8311 chc.727338 qd5.00391 ssch2558
6 conn3.2074 disc17.8489 pend.231637 cpu8056.23 u0
r0 10 t7 s3 16 lat780 d10 cr0_rps ro13.4774 la778 lap.9 latp.906299 laav436.65 i1270 e0 sz395.458 szs240.885 ti240861 th8.14514 io5.27276 ch146.228 chc3.95588 qd2.49843 ssch12101
conn68.3471 disc72.2224 pend2.70542 cpu2930.02 u0
r0 10 t7 s4 16 lat148 d10 cr0_rps ro.856389 la151 lap.899252 latp.89636 laav72.9165 i14190 e0 sz1.59027 szs.97415 ti64927.7 th1.35764 io218.551 ch30.1806 chc.736013 qd16.1677 ssch
14203 conn3.24539 disc26.1387 pend.251007 cpu8268.33 u0
r1 10 t7 s3 16 lat780 d20 cr1_rps ro12.6821 la788 lap.900069 latp.896377 laav470.234 i4333 e0 sz391.318 szs237.89 ti442493 th14.9683 io9.79224 ch149.276 chc3.41641 qd4.91369 ssch4
5200 conn67.4098 disc74.2216 pend4.70533 cpu11621.8 u0
r1 10 t7 s4 16 lat148 d20 cr1_rps ro1.02976 la148 lap.899919 latp.901434 laav73.933 i9951 e0 sz1.58688 szs.957054 ti21970 th2.80762 io452.935 ch28.94 chc1.04568 qd34.2326 ssch1098
1 conn3.24001 disc24.8488 pend.250638 cpu6376.45 u0
r1 10 t8 s3 16 lat780 d20 cr1_rps ro15.3699 la780 lap.900433 latp.900433 laav523.24 i693 e0 sz405.374 szs230.439 ti120664 th9.09437 io5.74324 ch161.472 chc4.30086 qd3.07504 ssch12
```

```
661 conn70.0218 disc84.9431 pend4.01129 cpu2988.66 u0
r1 10 t8 s4 16 lat148 d20 cr1_rps ro1.76362 la151 lap.899317 latp.895413 laav77.877 i13332 e0 sz1.59421 szs.984453 t162487.7 th1.32864 io213.354 ch30.2383 chc.889074 qd17.0644 ssc
h25392 conn3.24638 disc26.1807 pend.245807 cpu12355.5 u0
r5 10 t7 s3 16 lat780 d11 cr5_rps ro10.1128 la793 lap.899713 latp.891117 laav506.729 i698 e1858.29 sz405.537 szs231.835 ti120920 th9.14421 io5.77239 ch146.399 chc4.2817 qd2.98567
ssch7682 conn65.2225 disc76.025 pend2.46802 cpu3533.89 u0
r5 10 t7 s4 16 lat148 d11 cr5_rps ro.781576 la146 lap.899284 latp.905394 laav72.0881 i11484 e222.717 sz1.58177 szs.959935 t161066.6 th1.16196 io188.057 ch27.9394 chc.811588 qd13.8
148 ssch15049 conn3.23462 disc23.9189 pend.245878 cpu5738.15 u0
r5 10 t8 s3 16 lat780 d11 cr5_rps ro16.0217 la797 lap.900431 latp.892672 laav472.858 i2320 e68554.7 sz395.457 szs244.553 ti523840 th6.84146 io4.42883 ch115.449 chc2.82027 qd2.2435
3 ssch35582 conn52.7427 disc58.0988 pend2.24576 cpu68312.5 u0
r5 10 t8 s4 16 lat148 d11 cr5_rps ro2.02673 la148 lap.898715 latp.903599 laav89.5432 i7780 e1294.82 sz1.58342 szs.971368 ti120242 th.400203 io64.7031 ch24.12 chc.758659 qd5.87738
ssch28851 conn3.2358 disc20.0963 pend.251027 cpu9499.74 u0
r0 10 t7 s11 17 lat-1 d5 cr0_5 ro11.2365 la180 lap.899746 latp0 laav114.093 i788 e0 sz199.187 szs119.672 t190108.2 th6.80427 io8.74504 ch97.8557 chc4.03813 qd1.00127 ssch3768 conn
68.531 disc26.2164 pend.233376 cpu1021.05 u0
r0 10 t7 s11 17 lat720 d5 cr0_5 ro1.65872 la713 lap.899921 latp.903098 laav503.688 i1259 e0 sz197.061 szs118.693 ti120708 th8.02882 io10.4301 ch96.1804 chc4.64012 qd5.29627 ssch60
79 conn67.2675 disc25.7933 pend.230354 cpu1467.05 u0
r0 10 t7 s41 19 lat-1 d20 cr0_20 ro49.3459 la231 lap.899633 latp0 laav146.782 i817 e0 sz776.154 szs475.242 ti120300 th20.5903 io6.79134 ch96.1434 chc4.17927 qd1.00122 ssch15523 co
nn67.193 disc25.7781 pend.233497 cpu4367.56 u0
r0 10 t7 s41 19 lat924 d20 cr0_20 ro6.16034 la924 lap.9 latp.900781 laav693.815 i1280 e0 sz764.621 szs473.908 ti120929 th31.6144 io10.5847 ch95.598 chc5.058 qd7.44062 ssch24302 co
nn66.542 disc25.8244 pend.23154 cpu5192.47 u0
r0 10 t7 s4 110 lat148 d20 cr0_20 ro1.27936 la153 lap.900117 latp.892282 laav82.0494 i13707 e0 sz1.57584 szs.94497 t126900.3 th3.13658 io509.547 ch27.7637 chc1.04333 qd42.9815 ssc
h13735 conn3.22159 disc23.6927 pend.230996 cpu11184.6 u0
r0 10 t7 s4 110 lat148 d5 cr0_5 ro.799109 la150 lap.900089 latp.896827 laav76.5153 i26986 e0 sz1.57708 szs.94733 ti197483 th.841825 io136.65 ch27.8176 chc.614125 qd10.7778 ssch269
97 conn3.22366 disc23.8273 pend.230212 cpu15917.8 u0
r0 10 t7 s4 15 lat-1 d15 cr0_15 ro.827303 la37 lap.891619 latp0 laav28.1861 i3174 e0 sz1.57876 szs.981205 t190032.8 th.217412 io35.2538 ch27.134 chc.970004 qd1.00032 ssch3174 conn
3.22674 disc23.1624 pend.231723 cpu1384.63 u0
r0 10 t7 s4 15 lat148 d15 cr0_15 ro1.06837 la152 lap.900897 latp.895863 laav74.6596 i12738 e0 sz1.58431 szs.985128 t134353.5 th2.29473 io370.792 ch27.7693 chc.915595 qd28.2422 ssc
h12763 conn3.23579 disc23.7172 pend.230316 cpu9165.11 u0
r0 10 t7 s4 17 lat-1 d5 cr0_5 ro.530468 la37 lap.907581 latp0 laav27.9156 i4274 e0 sz1.62167 szs1.0251 ti120063 th.2255 io35.598 ch27.2557 chc.701713 qd1.00023 ssch4275 conn3.2906
disc23.2214 pend.231538 cpu1680.17 u0
r0 10 t7 s4 17 lat148 d5 cr0_5 ro.754723 la143 lap.899674 latp.909394 laav70.1113 i15027 e0 sz1.594 szs1.00739 ti118544 th.789298 io126.763 ch27.4869 chc.651365 qd9.00413 ssch1503
9 conn3.25145 disc23.4755 pend.230432 cpu8320.77 u0
r0 10 t7 s4 19 lat-1 d20 cr0_20 ro1.02513 la37 lap.890799 latp0 laav28.4825 i3141 e0 sz1.59726 szs1.01148 t190039.4 th.217657 io34.8847 ch27.2248 chc1.09595 qd1.00032 ssch3142 con
n3.25133 disc23.2217 pend.23282 cpu1447.99 u0
r0 10 t7 s4 19 lat148 d20 cr0_20 ro1.12144 la146 lap.899714 latp.904225 laav72.5909 i6709 e0 sz1.59144 szs.961717 ti14078.9 th2.96237 io476.527 ch27.8381 chc1.2698 qd35.7478 ssch6
733 conn3.24825 disc23.7443 pend.230506 cpu4722.06 u0
r0 10 t7 s8 15 lat-1 d15 cr0_15 ro37.3101 la219 lap.90137 latp0 laav137.013 i1095 e0 sz595.121 szs365.859 ti150453 th16.9192 io7.27804 ch97.7473 chc3.86817 qd1.00091 ssch15596 con
n68.7603 disc25.7511 pend.234316 cpu4232.05 u0
r0 10 t7 s8 15 lat876 d15 cr0_15 ro5.88047 la862 lap.902165 latp.907779 laav566.304 i1247 e0 sz580.484 szs353.55 ti120260 th23.5124 io10.3692 ch96.1939 chc5.01201 qd5.90778 ssch17
895 conn66.7347 disc26.1571 pend.231838 cpu3909.27 u0
r1 10 t7 s11 17 lat720 d10 cr1_5 ro3.50463 la719 lap.899688 latp.901471 laav451.993 i2244 e0 sz197.782 szs122.665 ti120529 th14.384 io18.618 ch94.2395 chc4.31234 qd8.47148 ssch119
28 conn67.776 disc23.3355 pend.230975 cpu3111.04 u0
r1 10 t7 s4 110 lat148 d20 cr1 ro1.15581 la145 lap.900511 latp.906803 laav76.0887 i7658 e0 sz1.58214 szs.964426 ti14033.9 th3.37241 io545.679 ch25.7455 chc1.43898 qd43.0837 ssch84
51 conn3.23232 disc21.6663 pend.229933 cpu5709.65 u0
r1 10 t7 s4 110 lat148 d10 cr1_5 ro.766301 la151 lap.900832 latp.8948 laav75.449 i8296 e0 sz1.58233 szs.980756 ti28319.7 th1.81066 io292.941 ch25.5761 chc.812798 qd22.3697 ssch913
5 conn3.22468 disc21.5467 pend.23019 cpu4407.63 u0
r1 10 t7 s4 17 lat148 d10 cr1_5 ro.885874 la142 lap.900004 latp.909398 laav66.8186 i24280 e0 sz1.58386 szs.96216 ti98903 th1.51885 io245.493 ch26.1913 chc.705472 qd16.6646 ssch268
03 conn3.23521 disc22.1691 pend.231038 cpu14826.6 u0
r1 10 t8 s11 17 lat720 d10 cr1_5 ro3.48483 la720 lap.899898 latp.900409 laav447.272 i1958 e0 sz196.629 szs120.303 ti180549 th8.32961 io10.8447 ch96.3563 chc3.44811 qd4.8907 ssch17
824 conn68.061 disc25.6645 pend.231275 cpu3969.25 u0
r1 10 t8 s4 110 lat148 d20 cr1 ro1.65207 la152 lap.899926 latp.89404 laav78.4763 i8190 e0 sz1.58974 szs.966947 t130939.5 th1.64383 io264.71 ch27.8979 chc.98308 qd21.2341 ssch15622
conn3.24293 disc23.8417 pend.23119 cpu7170.81 u0
r1 10 t8 s4 110 lat148 d10 cr1_5 ro1.11532 la148 lap.898898 latp.904476 laav78.7912 i7176 e0 sz1.57581 szs.998379 t149441.8 th.893411 io145.14 ch27.6812 chc.7271 qd11.5932 ssch136
17 conn3.21991 disc23.6934 pend.23093 cpu5199.95 u0
r1 10 t8 s4 17 lat148 d10 cr1_5 ro1.23726 la150 lap.899662 latp.897412 laav76.4706 i10668 e0 sz1.57415 szs.945771 t180890 th.810949 io131.883 ch27.6153 chc.719427 qd10.2518 ssch20
298 conn3.21872 disc23.6345 pend.231804 cpu8605.11 u0
r5 10 t7 s11 17 lat720 d6 cr5_5 ro2.35217 la713 lap.902786 latp.905882 laav482.412 i1615 e2561.96 sz196.672 szs114.946 ti150370 th8.25113 io10.7402 ch89.6619 chc4.04697 qd5.20557
ssch9595 conn61.3644 disc25.3161 pend.231721 cpu4485.08 u0
r5 10 t7 s41 19 lat924 d21 cr5_20 ro8.98857 la926 lap.900524 latp.896335 laav695.692 i956 e6902.48 sz822.302 szs489.902 ti100844 th30.4508 io9.47999 ch99.2061 chc5.38765 qd6.68933
ssch20111 conn67.0755 disc25.9632 pend.231617 cpu10160.8 u0
r5 10 t7 s4 110 lat148 d21 cr5_20 ro.912542 la152 lap.898977 latp.893101 laav76.8099 i17748 e377.11 sz1.58153 szs.968487 t141264.1 th2.65714 io430.108 ch26.15 chc.958191 qd33.6173
ssch23050 conn3.24099 disc22.0554 pend.23105 cpu8468.26 u0
r5 10 t7 s4 110 lat148 d6 cr5_5 ro.571704 la150 lap.901378 latp.898098 laav77.8041 i9157 e174.73 sz1.57781 szs.982083 t167451.8 th.836709 io135.756 ch26.0394 chc.671097 qd10.6295
ssch11895 conn3.23262 disc22.0493 pend.230292 cpu3821.24 u0
r5 10 t7 s4 15 lat148 d16 cr5_15 ro.879569 la147 lap.900783 latp.905402 laav73.4226 i17133 e339.44 sz1.57737 szs.945386 t155661.7 th1.89657 io307.806 ch26.0123 chc.880246 qd22.974
4 ssch22338 conn3.22951 disc21.9721 pend.230507 cpu8834.72 u0
```

```
r5 10 t7 s4 17 lat148 d6 cr5_5 ro.647409 la146 lap.90093 latp.909062 laav72.056 i12794 e227.564 sz1.58176 szs.958535 ti103708 th.762246 io123.366 ch25.9617 chc.659978 qd9.06784 ss
ch16505 conn3.21968 disc21.9796 pend.232122 cpu6043.37 u0
r5 10 t7 s4 19 lat148 d21 cr5_20 ro.967246 la146 lap.899966 latp.904724 laav73.0387 i8840 e194.758 sz1.58201 szs.986248 ti21924.3 th2.4917 io403.205 ch25.9588 chc1.24402 qd29.9109
ssch11623 conn3.23742 disc21.879 pend.230473 cpu4748.18 u0
r5 10 t7 s4 19 lat148 d20 cr5_19 ro.93677 la147 lap.899822 latp.902676 laav74.0598 i27069 e557.129 sz1.5855 szs.956773 ti69859.7 th2.39979 io387.477 ch26.0315 chc.928929 qd29.2609
ssch35059 conn3.24911 disc21.941 pend.230403 cpu14316.7 u0
r5 10 t7 s8 15 lat876 d16 cr5_15 ro6.07754 la881 lap.899745 latp.89719 laav671.535 i1567 e8457.98 sz585.886 szs348.465 ti150820 th23.7785 io10.3899 ch91.9465 chc4.50626 qd7.06892
ssch25128 conn63.4644 disc25.3114 pend.231075 cpu12639.7 u0
r5 10 t8 s11 17 lat720 d6 cr5_5 ro6.86114 la728 lap.900505 latp.895717 laav382.702 i3759 e52413.9 sz198.059 szs119.535 ti481245 th6.04313 io7.811 ch77.4232 chc2.87545 qd3.10694 ss
ch31681 conn52.7428 disc22.0913 pend.235653 cpu53967.3 u0
r5 10 t8 s41 19 lat924 d21 cr5_20 ro26.8281 la915 lap.899635 latp.902372 laav610.569 i1096 e73051.4 sz787.814 szs473.178 ti140708 th23.9704 io7.78918 ch78.7 chc4.15716 qd4.76551 s
sch31989 conn52.4603 disc21.5042 pend.23244 cpu67967.3 u0
r5 10 t8 s4 110 lat148 d21 cr5_20 ro2.94622 la151 lap.90023 latp.894262 laav91.8636 i6543 e1224.06 sz1.60813 szs.969601 ti45739.9 th.898593 io143.048 ch22.0368 chc1.08359 qd13.275
4 ssch24192 conn3.27013 disc17.917 pend.230333 cpu8331.05 u0
r5 10 t8 s4 110 lat148 d6 cr5_5 ro1.31611 la149 lap.901771 latp.901771 laav92.7061 i8979 e1437.4 sz1.58136 szs.951392 ti180233 th.30774 io49.8189 ch22.0387 chc.643815 qd4.6895 ssc
h33164 conn3.22705 disc18.0527 pend.232287 cpu9815.24 u0
r5 10 t8 s4 15 lat148 d16 cr5_15 ro2.49271 la149 lap.898913 latp.898913 laav87.5639 i57666 e9946.68 sz1.58823 szs.964906 ti570260 th.627366 io101.122 ch22.0217 chc.797074 qd9.4729
3 ssch213208 conn3.23902 disc17.9759 pend.231492 cpu72937.4 u0
r5 10 t8 s4 17 lat148 d6 cr5_5 ro1.44141 la146 lap.898521 latp.912336 laav86.7338 i8179 e1339.62 sz1.58173 szs.950808 ti180126 th.280555 io45.4072 ch22.0311 chc.653343 qd4.11407 s
sch30350 conn3.23257 disc18.0432 pend.232104 cpu9431.19 u0
r5 10 t8 s4 19 lat148 d21 cr5_20 ro3.01312 la148 lap.899211 latp.902753 laav88.8523 i6216 e1129.82 sz1.57883 szs.949783 ti45708.9 th.838698 io135.991 ch22.0465 chc1.07097 qd12.231
7 ssch23040 conn3.22259 disc17.9822 pend.230767 cpu8238.61 u0
r5 10 t8 s8 15 lat876 d16 cr5_15 ro13.7673 la881 lap.900058 latp.895991 laav603.941 i1721 e82953.9 sz581.9 szs359.271 ti210881 th18.5504 io8.16102 ch76.5043 chc3.76828 qd4.95352 s
sch38170 conn51.9973 disc21.7752 pend.232808 cpu76307.8 u0
r0 10 t13 s3 110 lat-1 d10 cr0 ro24.1547 la197 lap.901981 latp0 laav125.156 i959 e0 sz394.277 szs228.843 ti120280 th12.2797 io7.97309 ch96.8582 chc3.84001 qd1.00104 ssch9188 conn
67.6977 disc26.0028 pend.233278 cpu2362.57 u0
r0 10 t13 s3 110 lat788 d10 cr0 ro3.64949 la791 lap.899764 latp.899764 laav573.842 i848 e0 sz386.82 szs233.658 ti80755.4 th15.8669 io10.5008 ch95.4764 chc5.65818 qd6.11439 ssch81
06 conn66.6576 disc25.5711 pend.232361 cpu1728.19 u0
r0 10 t13 s4 110 lat-1 d10 cr0 ro.657311 la37 lap.911594 latp0 laav27.9834 i5339 e0 sz1.57558 szs.981974 ti150079 th.218947 io35.5745 ch27.1271 chc.810185 qd1.00019 ssch5340 conn
3.22383 disc23.1584 pend.230352 cpu1974.33 u0
r0 10 t13 s4 110 lat148 d10 cr0 ro.921574 la148 lap.90034 latp.901735 laav74.3227 i16502 e0 sz1.58314 szs.954594 ti62530.4 th1.63202 io263.904 ch27.706 chc.736584 qd19.9243 ssch1
6519 conn3.23453 disc23.6834 pend.230716 cpu10148.2 u0
r0 10 t7 s29 16 lat-1 d10 cr0 ro3.87013 la58 lap.902459 latp0 laav48.9863 i2440 e0 sz36.1324 szs20.8056 ti120102 th2.86745 io20.3161 ch40.2018 chc1.34676 qd1.00041 ssch8962 conn1
5.9099 disc23.031 pend.253072 cpu2703.23 u0
r0 10 t7 s29 16 lat232 d10 cr0 ro2.17502 la231 lap.901122 latp.908722 laav131.007 i13817 e0 sz36.3502 szs20.7552 ti270288 th7.25862 io51.1195 ch40.0181 chc1.17305 qd6.96562 ssch5
1045 conn15.9164 disc22.8244 pend.244746 cpu14127.3 u0
r0 10 t7 s31 16 lat-1 d10 cr0 ro11.4571 la96 lap.89892 latp0 laav69.2118 i1296 e0 sz134.397 szs78.3445 ti90083.3 th7.55282 io14.3867 ch51.3217 chc1.86782 qd1.00077 ssch10886 conn
26.0891 disc23.5962 pend.247275 cpu2891.87 u0
r0 10 t7 s31 16 lat384 d10 cr0 ro3.28456 la387 lap.899878 latp.899355 laav225.498 i5734 e0 sz135.588 szs82.2777 ti270569 th11.2243 io21.1924 ch51.4578 chc1.5999 qd4.8556 ssch4775
2 conn26.5622 disc23.2314 pend.235162 cpu10920.8 u0
r0 10 t7 s32 16 lat-1 d10 cr0 ro34.1177 la285 lap.900289 latp0 laav173.793 i692 e0 sz625.069 szs346.822 ti120497 th14.0223 io5.74288 ch137.605 chc6.07034 qd1.00145 ssch6795 conn1
04.922 disc28.2079 pend.231869 cpu1802.68 u0
r0 10 t7 s32 16 lat1140 d10 cr0 ro4.12009 la1143 lap.901361 latp.897959 laav770.7 i882 e0 sz606.287 szs361.784 ti120747 th17.2994 io7.30456 ch135.607 chc7.49206 qd5.72109 ssch864
0 conn102.626 disc28.7509 pend.231985 cpu1903.84 u0
r0 10 t7 s42 16 lat-1 d10 cr0 ro17.7719 la141 lap.89905 latp0 laav94.8532 i842 e0 sz257.48 szs149.159 ti80145.8 th10.5666 io10.5058 ch71.8674 chc2.45907 qd1.00119 ssch7861 conn45
.2567 disc24.2773 pend.235598 cpu2059.14 u0
r0 10 t7 s42 16 lat564 d10 cr0 ro3.64391 la565 lap.9 latp.9 laav420.17 i1130 e0 sz254.466 szs155.822 ti80296.5 th13.9885 io14.0728 ch71.9261 chc3.26208 qd5.98053 ssch10470 conn45
.1323 disc24.4134 pend.232662 cpu2339.99 u0
r1 10 t13 s3 110 lat788 d20 cr1 ro7.35353 la795 lap.899757 latp.893917 laav514.663 i2055 e0 sz395.247 szs246.818 ti100684 th31.5124 io20.4104 ch94.5148 chc4.78664 qd10.5985 ssch1
9628 conn68.1244 disc22.8789 pend.231043 cpu4606.16 u0
r1 10 t13 s4 110 lat148 d20 cr1 ro.969344 la151 lap.900531 latp.897212 laav79.2317 i9093 e0 sz1.5788 szs.939662 ti13977.5 th4.01204 io650.548 ch24.7844 chc1.10495 qd53.2063 ssch9
128 conn3.22619 disc20.6995 pend.230198 cpu5842.51 u0
r1 10 t7 s29 16 lat232 d20 cr1 ro3.06421 la231 lap.899987 latp.904119 laav127.527 i15260 e0 sz36.3325 szs20.6334 ti150178 th14.4213 io101.613 ch37.9218 chc1.26953 qd13.205 ssch62
175 conn15.916 disc20.6744 pend.233653 cpu18256 u0
r1 10 t7 s31 16 lat384 d20 cr1 ro4.42954 la390 lap.899823 latp.89516 laav258.036 i6221 e0 sz133.754 szs80.8085 ti150385 th21.6133 io41.3671 ch48.4263 chc1.79233 qd10.7635 ssch567
68 conn26.2967 disc20.4106 pend.232316 cpu13850.4 u0
r1 10 t7 s32 16 lat1140 d20 cr1 ro8.98049 la1155 lap.900037 latp.893673 laav757.148 i2676 e0 sz626.499 szs356.164 ti210702 th31.0811 io12.7004 ch137.226 chc5.71585 qd9.69507 ssch
28629 conn105.632 disc26.7549 pend.231195 cpu6882.13 u0
r1 10 t7 s42 16 lat564 d20 cr1 ro5.97754 la562 lap.899277 latp.90506 laav382.237 i2076 e0 sz251.758 szs148.742 ti80571.4 th25.339 io25.766 ch69.234 chc3.50043 qd9.97832 ssch21509
conn44.892 disc21.9054 pend.231732 cpu5189.77 u0
r1 10 t8 s29 16 lat232 d20 cr1 ro4.31474 la235 lap.899897 latp.894758 laav143.67 i4865 e0 sz36.8602 szs20.8721 ti90140 th7.77111 io53.9716 ch39.8685 chc1.34538 qd7.81768 ssch3477
4 conn15.9183 disc22.7539 pend.232008 cpu9092.33 u0
r1 10 t8 s31 16 lat384 d20 cr1 ro5.3016 la381 lap.897765 latp.911555 laav274.772 i2104 e0 sz129.16 szs78.5525 ti90237.3 th11.7638 io23.3163 ch50.4872 chc1.80293 qd6.4653 ssch3295
2 conn25.6481 disc23.3169 pend.233311 cpu7416.16 u0
r1 10 t8 s32 16 lat1140 d20 cr1 ro9.01628 la1131 lap.901189 latp.905764 laav764.642 i1093 e0 sz640.58 szs367.82 ti150785 th18.1383 io7.24874 ch141.449 chc6.70374 qd5.59469 ssch20
```

```
387 conn108.418 disc28.8446 pend.230559 cpu4521.42 u0
 r1 10 t9 s3 110 lat788 d20 cr1 ro6.94645 la791 lap.900358 latp.897971 laav566.263 i1676 e0 sz392.589 szs245.939 ti160549 th16.009 io10.4392 ch96.4786 chc3.74037 qd5.95227 ssch317
67 conn68.246 disc25.4102 pend.231602 cpu6728.31 u0
 r1 10 t9 s4 110 lat148 d20 cr1 ro1.5812 la151 lap.901287 latp.895945 laav77.9616 i8254 e0 sz1.57487 szs.933144 ti32337.6 th1.57023 io255.245 ch27.9017 chc1.00846 qd20.4351 ssch16
534 conn3.21909 disc23.8778 pend.230321 cpu6881.83 u0
 r5 10 t13 s3 110 lat788 d11 cr5 ro4.09699 la791 lap.899763 latp.89739 laav572.946 i1686 e0 sz402.837 szs240.666 ti160844 th16.4946 io10.4822 ch95.5716 chc4.11441 qd6.0427 ssch176
58 conn66.9424 disc25.5324 pend.231412 cpu4060.99 u0
 r5 10 t13 s4 110 lat148 d11 cr5 ro.607716 la143 lap.899363 latp.910124 laav71.8055 i11635 e0 sz1.59424 szs.983203 ti40526.3 th1.7879 io287.098 ch27.7313 chc.807752 qd20.8077 ssch
11656 conn3.25182 disc23.6965 pend.230852 cpu4136.22 u0
 r5 10 t7 s29 16 lat232 d11 cr5 ro2.85085 la231 lap.898946 latp.904819 laav122.95 i18219 e9131.75 sz36.9497 szs20.9164 ti420315 th6.25634 io43.346 ch39.5144 chc1.19834 qd5.84302 s
sch78002 conn16.0085 disc22.2237 pend.236517 cpu29700.7 u0
 r5 10 t7 s31 16 lat384 d11 cr5 ro3.084 la381 lap.900746 latp.908955 laav275.362 i2680 e3841.76 sz132.169 szs79.422 ti120246 th11.5066 io22.2876 ch49.4233 chc1.82597 qd6.18918 ssc
h24665 conn24.9897 disc22.8157 pend.233534 cpu8803.53 u0
 r5 10 t7 s32 16 lat1140 d11 cr5 ro4.4191 la1143 lap.90005 latp.896072 laav783.97 i2011 e10135.1 sz604.968 szs350.04 ti271431 th17.5083 io7.40888 ch128.548 chc5.56605 qd5.84286 ss
ch22465 conn96.665 disc27.6126 pend.231636 cpu13803.9 u0
 r5 10 t7 s42 16 lat564 d11 cr5 ro3.55574 la563 lap.903061 latp.911565 laav421.372 i1177 e2495.67 sz252.121 szs157.685 ti80406.1 th14.4164 io14.6382 ch68.9087 chc3.48147 qd6.24979
 ssch12150 conn42.6303 disc23.996 pend.232749 cpu4847.04 u0
 r5 10 t8 s29 16 lat232 d11 cr5 ro5.63128 la230 lap.900372 latp.909171 laav129.848 i10228 e44752.1 sz37.1185 szs20.9776 ti540666 th2.74291 io18.9174 ch37.7292 chc1.25717 qd2.90673
 ssch86727 conn16.3337 disc20.0957 pend.238739 cpu62857.3 u0
 r5 10 t8 s31 16 lat384 d11 cr5 ro9.77173 la381 lap.900314 latp.903552 laav194.736 i9881 e113242 sz133.603 szs79.0477 ti811541 th6.35428 io12.1756 ch47.0787 chc1.55016 qd2.61036 s
sch134219 conn24.5318 disc20.942 pend.238007 cpu132789 u0
 r5 10 t8 s32 16 lat1140 d11 cr5 ro10.0212 la1151 lap.900133 latp.890812 laav817.735 i751 e33267.8 sz588.465 szs339.585 ti120460 th14.331 io6.23441 ch100.93 chc6.84744 qd5.14647 s
sch11774 conn74.152 disc23.3471 pend.232159 cpu30616.6 u0
 r5 10 t9 s3 110 lat788 d11 cr5 ro5.98854 la764 lap.900366 latp.904936 laav541.84 i1094 e35558.4 sz393.229 szs241.965 ti120518 th13.9435 io9.07748 ch82.7963 chc4.18272 qd4.95247 s
sch14783 conn56.8818 disc23.0959 pend.232085 cpu32813.9 u0
 r5 10 t9 s4 110 lat148 d11 cr5 ro2.06456 la149 lap.902746 latp.902746 laav90.6663 i10634 e1954.09 sz1.5773 szs.932404 ti148196 th.442113 io71.7562 ch21.8322 chc.753286 qd6.69466
 ssch42560 conn3.22528 disc17.8261 pend.231086 cpu13401.3 u0
r0 1 t7 s29 16 lat-1 d10 cr0 ro7.73289 la56 lap.902314 latp0 laav46.0055 i1945 e0 sz33.2889 szs19.669 ti90089.5 th2.80741 io21.5896 ch30.5946 chc.916292 qd1.00051 ssch18477 conn6.
44455 disc23.2542 pend.264611 cpu4825.3 u0
r0 1 t7 s29 16 lat232 d10 cr0 ro2.55908 la233 lap.901587 latp.901587 laav139.956 i8505 e0 sz32.9405 szs19.925 ti270268 th4.04921 io31.4688 ch30.8584 chc.769077 qd4.44868 ssch80220
 conn6.43139 disc23.5296 pend.234319 cpu18134.4 u0
r0 1 t7 s3 16 lat-1 d10 cr0 ro23.9002 la196 lap.899257 latp0 laav123.794 i1211 e0 sz387.073 szs231.863 ti150318 th12.1811 io8.05625 ch95.113 chc3.54668 qd1.00083 ssch12043 conn64.
9685 disc27.1368 pend.234445 cpu3143.29 u0
r0 1 t7 s3 16 lat780 d10 cr0 ro3.02699 la779 lap.900747 latp.901814 laav609.637 i937 e0 sz377.909 szs237.067 ti90399 th15.3011 io10.3652 ch93.1064 chc5.38106 qd6.42263 ssch9357 co
nn63.3425 disc26.7515 pend.23202 cpu2031.33 u0
r0 1 t7 s4 16 lat-1 d10 cr0 ro1.15352 la39 lap.909317 latp0 laav29.7152 i4014 e0 sz1.60438 szs.964977 ti120068 th.209516 io33.431 ch26.3915 chc.783133 qd1.00025 ssch6440 conn2.379
07 disc23.2915 pend.2368 cpu2317.86 u0
r0 1 t7 s4 16 lat156 d10 cr0 ro.928234 la161 lap.900923 latp.894167 laav78.8866 i39233 e0 sz1.58879 szs.962066 ti240083 th1.01418 io163.414 ch26.4914 chc.628824 qd13.5317 ssch6234
4 conn2.37875 disc23.3634 pend.232461 cpu23352.1 u0
r1 1 t7 s29 16 lat232 d20 cr1 ro4.44936 la231 lap.899166 latp.903069 laav123.014 i16909 e0 sz32.6747 szs19.9487 ti300255 th7.18785 io56.3155 ch29.4699 chc.810805 qd7.307 ssch17495
7 conn6.38933 disc22.133 pend.234056 cpu43507.4 u0
r1 1 t7 s3 16 lat780 d20 cr1 ro6.88464 la779 lap.900589 latp.901325 laav515.895 i2716 e0 sz384.149 szs239.659 ti150829 th27.0214 io18.0072 ch93.2314 chc3.67827 qd9.39065 ssch30142
 conn64.5031 disc25.6089 pend.231756 cpu6983.47 u0
r1 1 t7 s4 16 lat148 d20 cr1 ro1.22001 la151 lap.90007 latp.896895 laav75.3431 i97974 e0 sz1.5811 szs.960012 ti294089 th2.05756 io333.144 ch25.0773 chc.724723 qd26.2987 ssch30142
conn64.5031 disc25.6089 pend.231756 cpu68444.5 u1
r1 1 t8 s29 16 lat232 d20 cr1 ro5.93334 la234 lap.898673 latp.895095 laav128.842 i8666 e0 sz32.7045 szs19.8944 ti270167 th4.09783 io32.0765 ch30.9715 chc.786318 qd4.26321 ssch1543
71 conn6.40718 disc23.6597 pend.233436 cpu34952.5 u0
r1 1 t8 s3 16 lat780 d20 cr1 ro7.20171 la778 lap.899789 latp.901547 laav470.597 i2844 e0 sz391.748 szs239.674 ti270528 th16.0873 io10.5128 ch95.4276 chc3.04602 qd5.01653 ssch54099
 conn65.7574 disc26.8727 pend.231452 cpu11690.8 u0
r1 1 t8 s4 16 lat148 d20 cr1 ro1.92286 la148 lap.899743 latp.90199 laav80.2482 i15595 e0 sz1.57961 szs.951869 ti90092.3 th1.06809 io173.1 ch26.5941 chc.793674 qd13.9965 ssch46889
conn2.37917 disc23.4471 pend.230866 cpu14644.3 u0
r5 1 t7 s29 16 lat232 d11 cr5 ro2.43385 la230 lap.901062 latp.913351 laav169.638 i4801 e1172.91 sz32.7955 szs19.8368 ti150233 th4.09393 io31.957 ch29.7166 chc.806434 qd5.46095 ssc
h51725 conn6.09009 disc22.7391 pend.233738 cpu13068 u0
r5 1 t7 s3 16 lat780 d11 cr5 ro4.21554 la793 lap.899001 latp.891232 laav482.898 i2703 e6329.08 sz394.772 szs244.064 ti270698 th15.3981 io9.9853 ch92.3669 chc3.6301 qd4.88383 ssch3
1074 conn62.9252 disc26.4083 pend.231627 cpu17177.5 u0
r5 1 t7 s4 16 lat148 d11 cr5 ro.99488 la148 lap.90088 latp.902434 laav78.2195 i32176 e624.428 sz1.59286 szs.968013 ti210141 th.952709 io153.116 ch25.1121 chc.642416 qd12.2544 ssch
63102 conn2.37966 disc21.985 pend.231973 cpu20688.6 u0
r5 1 t8 s29 16 lat232 d11 cr5 ro6.51337 la235 lap.90095 latp.896814 laav122.764 i8945 e18019.2 sz32.6714 szs19.862 ti420599 th2.71419 io21.2673 ch26.2734 chc.786277 qd2.84919 ssch
131494 conn5.55325 disc19.8511 pend.242029 cpu48603.3 u0
r5 1 t8 s3 16 lat780 d11 cr5 ro6.33618 la785 lap.900138 latp.897837 laav531.504 i2173 e50430.4 sz393.314 szs234.853 ti241076 th13.8486 io9.01375 ch70.3827 chc2.8312 qd4.84768 ssch
34738 conn46.0647 disc21.9238 pend.231777 cpu53704.4 u0
r5 1 t8 s4 16 lat148 d11 cr5 ro2.25727 la147 lap.898938 latp.904248 laav92.5055 i11490 e1960.69 sz1.60139 szs.989694 ti180140 th.398994 io63.7836 ch21.63 chc.683325 qd5.96084 ssch
56319 conn2.38635 disc18.5007 pend.23242 cpu17063.2 u0
r1 10 t10 s3 16 lat780 d20 cr1 ro7.38959 la779 lap.900877 latp.906945 laav559.419 i1483 e0 sz383.635 szs230.499 ti120841 th18.3909 io12.2723 ch94.9682 chc4.12938 qd6.94268 ssch239
48 conn66.3359 disc25.7914 pend.231777 cpu5374.19 u0
```

```
r1 10 t10 s4 16 lat148 d20 cr1 ro1.37462 la153 lap.900629 latp.893888 laav76.7683 i24657 e0 sz1.59302 szs.983642 ti87676.7 th1.74999 io281.226 ch27.7173 chc.85931 qd22.0134 ssch41
890 conn3.24669 disc23.6539 pend.230232 cpu17987.5 u0
r1 10 t11 s3 16 lat780 d20 cr1 ro9.70465 la796 lap.900578 latp.89191 laav446.498 i5884 e0 sz392.544 szs240.838 ti451104 th20.0006 io13.0435 ch96.8005 chc3.04431 qd5.94697 ssch8444
7 conn68.2936 disc25.4802 pend.231445 cpu19720.6 u0
r1 10 t11 s4 16 lat148 d20 cr1 ro1.13477 la149 lap.899982 latp.899982 laav73.7727 i5582 e0 sz1.58259 szs.948459 ti16584.2 th2.08076 io336.585 ch27.3734 chc1.28574 qd25.3432 ssch83
29 conn3.23636 disc23.3269 pend.228891 cpu3548.17 u0
r1 10 t12 s3 16 lat780 d20 cr1 ro8.81903 la795 lap.900541 latp.89027 laav477.84 i1850 e0 sz390.129 szs237.911 ti120960 th23.3076 io15.2943 ch95.2072 chc3.95255 qd7.46649 ssch23215
 conn67.3031 disc24.7709 pend.231624 cpu5427.43 u0
r1 10 t12 s4 16 lat148 d20 cr1 ro1.05331 la143 lap.900764 latp.910519 laav70.4278 i20852 e0 sz1.58316 szs.953423 ti52768.8 th2.44374 io395.158 ch27.0144 chc.90114 qd28.4276 ssch27
072 conn3.22938 disc22.9556 pend.2307 cpu12668.2 u0
r1 10 t7 s3 16 lat780 d20 cr1 ro7.31506 la777 lap.899642 latp.903823 laav526.545 i3351 e0 sz393.677 szs233.723 ti180860 th28.4926 io18.5281 ch94.6119 chc4.21389 qd9.86273 ssch3544
3 conn67.6844 disc23.6592 pend.231619 cpu8428.74 u0
r1 10 t7 s4 16 lat148 d20 cr1 ro1.14824 la142 lap.900116 latp.910812 laav70.8236 i13874 e0 sz1.59478 szs.986644 ti27662.8 th3.1244 io501.539 ch25.9125 chc1.08618 qd36.3416 ssch153
11 conn3.26141 disc21.7997 pend.230979 cpu10319.1 u0
r1 10 t8 s3 16 lat780 d20 cr1 ro7.0149 la794 lap.900815 latp.890285 laav509.254 i2944 e0 sz390.914 szs236.423 ti271229 th16.5745 io10.8543 ch96.178 chc3.15155 qd5.59952 ssch53553
conn67.3689 disc25.9602 pend.231518 cpu11779.7 u0
r1 10 t8 s4 16 lat148 d20 cr1 ro1.70633 la151 lap.899304 latp.894863 laav77.6186 i8383 e0 sz1.58833 szs.944376 ti33757.5 th1.54074 io248.33 ch27.9492 chc1.10145 qd19.7613 ssch1599
9 conn3.23857 disc23.8968 pend.230574 cpu7835.77 u0
r5 10 t10 s3 16 lat780 d11 cr5 ro7.41159 la793 lap.899746 latp.892766 laav577.572 i1576 e38216.7 sz390.728 szs233.813 ti181052 th13.2858 io8.70468 ch78.6607 chc3.74227 qd5.07931 s
sch22771 conn53.7901 disc22.1997 pend.232127 cpu37084.2 u0
r5 10 t10 s4 16 lat148 d11 cr5 ro1.7003 la145 lap.900766 latp.910573 laav79.2716 i28351 e3672.57 sz1.58023 szs.96712 ti351994 th.497178 io80.544 ch22.4552 chc.716772 qd7.10811 ssc
h88009 conn3.22789 disc18.4534 pend.230932 cpu29665.7 u0
r5 10 t11 s3 16 lat780 d11 cr5 ro6.20801 la793 lap.900777 latp.892409 laav535.168 i1675 e28253.4 sz394.983 szs245.341 ti180675 th14.3039 io9.27078 ch84.099 chc4.0096 qd4.99045 ssc
h21837 conn57.6201 disc23.5878 pend.231985 cpu28553.8 u0
r5 10 t11 s4 16 lat148 d11 cr5 ro1.31896 la144 lap.8999 latp.913686 laav77.5995 i10015 e919.557 sz1.58233 szs.957294 ti86460.1 th.715964 io115.834 ch23.0894 chc.758186 qd9.09446 s
sch24891 conn3.23277 disc19.0679 pend.231347 cpu8543.07 u0
r5 10 t12 s3 16 lat780 d11 cr5 ro6.06121 la790 lap.899854 latp.896451 laav475.538 i2057 e20464.9 sz395.77 szs234.019 ti210776 th15.0875 io9.75917 ch87.389 chc3.71401 qd4.66845 ssc
h24749 conn60.3137 disc24.1277 pend.232468 cpu22773.2 u0
r5 10 t12 s4 16 lat148 d11 cr5 ro1.0421 la145 lap.900354 latp.908641 laav73.5337 i24869 e1357.6 sz1.57602 szs.951592 ti171062 th.895004 io145.38 ch23.9855 chc.710389 qd11.3079 ssc
h47011 conn3.20936 disc19.9964 pend.23104 cpu16743.2 u0
r5 10 t7 s3 16 lat780 d11 cr5 ro4.07356 la779 lap.900552 latp.90292 laav579.534 i1267 e4451.46 sz391.711 szs234.774 ti120693 th16.0627 io10.4977 ch91.55 chc4.78054 qd6.16101 ssch1
3896 conn63.2224 disc25.2504 pend.231166 cpu6859.24 u0
r5 10 t7 s4 16 lat148 d11 cr5 ro.744581 la152 lap.899452 latp.894407 laav76.7026 i22809 e437.237 sz1.57565 szs.969248 ti101909 th1.37756 io223.816 ch25.8772 chc.714765 qd17.4603 s
sch29782 conn3.21999 disc21.8715 pend.231446 cpu11274.2 u0
r5 10 t8 s3 16 lat780 d11 cr5 ro8.26841 la785 lap.9 latp.899444 laav556.377 i1800 e57045.7 sz401.173 szs247.472 ti220911 th12.7687 io8.14809 ch77.9747 chc3.34912 qd4.565 ssch27790
 conn53.2878 disc22.0059 pend.232394 cpu54164.3 u0
r5 10 t8 s4 16 lat148 d11 cr5 ro1.94845 la149 lap.9 latp.9 laav87.5946 i20341 e3368.52 sz1.57426 szs.966197 ti277651 th.450515 io73.261 ch22.0298 chc.720801 qd6.82031 ssch75215 co
nn3.22321 disc18.0324 pend.230877 cpu24776.2 u0
```