Latent Frequency Synthesis for Behavioral Hard Disk Drive Access Time Models

Adam Crume
University of California
Santa Cruz, California
adamcrume@cs.ucsc.edu

Carlos Maltzahn
University of California
Santa Cruz, California
carlosm@cs.ucsc.edu

ABSTRACT

Access time prediction is important for predicting hard disk drive latencies. Existing approaches use white-box models developed with expert knowledge and can take months to parameterize. Automatically learning behavior is much preferred, requiring less expertise, fewer assumptions, and less time. Black-box modeling has been attempted, but none have demonstrated per-request accuracy. Barriers to machine learning of access times include periodicity with high, unknown frequencies, and a discontinuous access time function. Previously, we showed that recognizing periodicity is crucial for accuracy. Unfortunately, Fourier analysis is expensive and ill-suited for this problem. In this paper, we demonstrate that the model can identify periods itself, bypassing Fourier analysis. We show that neural nets, given specific sinusoids as additional inputs, can predict access times, even across differing track lengths and thus differing periods. This removes a significant barrier and expands the range, although challenges remain. Removing discontinuities via a trigonometric transformation further reduces error.

Categories and Subject Descriptors

B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*; D.4.8 [**Operating Systems**]: Performance—*Modeling and predictions*

General Terms

Performance

Keywords

neural net, periodicity, Fourier analysis, hard disk drive

1. INTRODUCTION

In an earlier paper, we showed that recognition of periodicity is crucial for predicting hard disk drive access times [11]. A neural net could be used to predict hard disk drive access times, but it only worked if the neural net was explicitly fed periodicity information. This consisted of adding sines and cosines of the block numbers of various periods as additional inputs. Results were best when using periods detected through Fourier analysis. Unfortunately, even with the Fourier analysis constrained to one dimension, the computational cost of scaling up to the billions of sectors on modern hard disk drives is prohibitive. Furthermore, effective functioning of the neural net may require periods not found by Fourier analysis (fig. 3).

The issues associated with Fourier analysis can be avoided by synthesizing the sinusoids in the neural net itself. To do this, a base set of sinusoids with frequencies which are powers of two are fed into the neural net. Just as any positive integer can be constructed by adding a logarithmic number of powers of two, sinusoids with positive integer frequencies can be constructed by multiplying a logarithmic number of complex sinusoids with frequencies which are powers of two. The neural net is also effective at constructing sinusoids with non-integer frequencies, presumably via nonlinear behavior (see fig. 1). Since the neural net is able to construct sinusoids with whatever periods are necessary, the issue of periodic behavior not showing up in Fourier analysis is moot.

Hard disk drive latency models can be used when simulating large compute systems [35], for parallel file system simulation [36], for storage configuration [4], and for data placement [25]. The most accurate models are white-box models, which require detailed knowledge of the device's internals. One such example of a white-box model is DiskSim [8], which is widely used [9, 55, 36]. These models have many configuration parameters which must be precisely determined in order for the model to be accurate. This task of parameterization can take a long time.

Other researchers have noted the difficulty of parameterizing DiskSim [40, 32]. Many researchers use the outdated disk models that come with DiskSim, presumably because this is easier than obtaining or creating models for newer disk drives. Hard disk drives commonly modeled with DiskSim include the Quantum Atlas 10K [12, 49, 31, 53], which was released in 1999 [31]; the Seagate Cheetah 9LP ST39102LW [49, 33, 50, 34], which appears to have been released in 1998 [38]; and the Seagate Barracuda 4LP ST32171 [55], which appears to have been released between 1996 and 1998 [5]. All of these papers used models which come with DiskSim, and these models were at least 11 years old when the papers

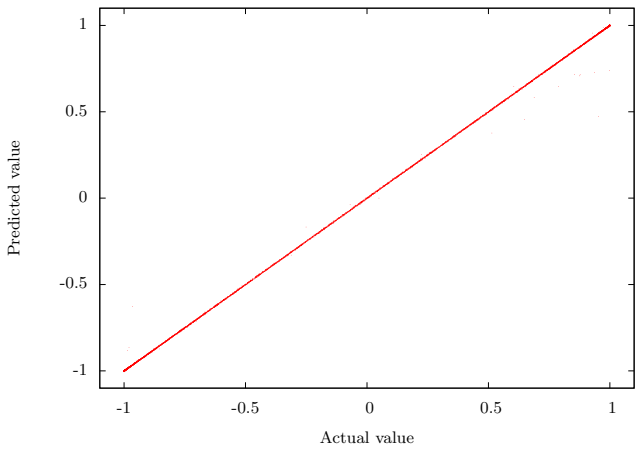


Figure 1: Predicted versus actual value of $\sin(1000.47 \cdot 2\pi x)$ generated by a neural net. The neural net has 4 hidden layers of 100 neurons each. The input consists of sines and cosines of x with frequencies of the form 2^k with $0 \leq k < 30$. The mean absolute error* is 0.00215. This shows that the neural net can generate non-integer frequencies well.
*We use mean absolute error because our errors are very non-Gaussian.

were published. The capacities of the modeled devices were at most 9.2 GB, with sustained data transfer rates of at most 41 MB/s [1, 5, 38]. For comparison, consider top of the line models released in 2010, the year of the earliest publication above. The WD Caviar Green 3TB [13] and the Seagate 3TB FreeAgent GoFlex Desk [2], had capacities of around 3000 GB and sustained data transfer rates of around 123 MB/s [3] (sizes measured using $1\,\text{GB} = 10^9$ bytes). This means that the models used in the papers were at least a factor of 326 times smaller and 3 times slower than top of the line drives of the time. Using such out of date models likely hampers research relevance, but it is nearly inevitable when models are so hard to create.

Parameterizing white-box models is a long and difficult process, because manufacturers do not release details such as sector layout required by white-box models. In fact, a fellow researcher (Ron Oldfield) was involved with configuring DiskSim to model an existing device, a process that took several months [41]. Tools such as DIG can extract some of this information, such as the sector layout [22]. Unfortunately, they require assumptions about the internal structure of the hard disk drive. This structure is likely to change in the future due to the introduction of shingled hard disk drives [45], dual-heads per surface [31], or other optimizations, as has happened in the past with Zoned Bit Recording and serpentine layouts. Since manufacturers do not release this information, researchers are forced to reverse-engineer a device before modifying DIG and DiskSim to support the new layout.

Machine learning is a much more desirable approach than white box modeling for hard disk drive latency prediction. *Behavioral models* requiring a minimal set of assumptions can be constructed using machine learning. This stands in stark contrast to white-box modeling, which requires intimate knowledge of the device's internals. Researchers have had success with behavioral modeling of hard disk drive aggregate performance [27, 12, 51, 55], but none of these can accurately model individual requests.

In this paper, we use a workload consisting of random, single-sector reads. The latencies of this workload are known to be difficult to predict on a per-request level [11]. They also correspond directly to access times, which are an important part of request latencies in general, so this is a problem worth addressing. At the same time, this workload reduces the impact of complex firmware behavior such as caching and readahead, making the problem tractable. We see this as a stepping stone toward a performance simulator for more general workloads.

The main contribution of this paper is to show the ability to recognize periodicity based on a generic base set of sines and cosines, rather than sines and cosines with periods extracted from the data. We also show that discontinuities in the access time function are the main source of remaining error and how they can be removed.

2. RELATED WORK

2.1 Predicting request latencies

DiskSim [8] is a well-regarded disk model based on discrete event simulation. It has been validated to produce request-level accuracy. However, it is computationally expensive and difficult to configure for modern disks.

Many analytic models exist, including work by Lebrecht, Dingle, and Knottenbelt [30]. Analytic models are relatively easy to understand and extremely fast due to their compact formulae. Unfortunately, they require very detailed expert knowledge to create. Often, they are limited to certain classes of workloads and are not useful alone in generalized contexts. Our approach shares the last two limitations (for now) but does not require domain expertise.

Kelly *et al.* describe a black-box probabilistic model [27] similar to table-based models such as Garcia *et al.* [19]. In this approach, requests are categorized based on features including size, LRU stack distance, number of pending reads, number of pending writes, and some RAID-specific information. Table-based models are limited by the table size. The work by Kelly *et al.* ameliorates this issue by essentially not requiring the entire table to be filled in, but the problem of table size is not fully solved.

Mesnier *et al.* create a model for relative performance of storage devices [37]. Unfortunately, their approach still requires an accurate base model.

Using regression trees to predict response time is a popular approach. Dai *et al.* predict performance with a combination of regression trees and support vector regression [12]. However, their models are workload-specific, and their prediction errors are based on one-second averages rather than per-request latencies. Wang *et al.* also calculate errors based on windows, using one-minute averages in their case [51].

None of the machine-learning-based approaches have shown

low per-request errors.

2.2 Learning periodic functions

Neural nets can be trained on periodic functions in many ways. Some setups predict the value of the function directly from the input. These invariably use a fixed interval with a very small number of oscillations [24, 23, 43, 18, 54]. With this approach, extrapolating beyond the training range leads to poor performance [28]. This approach is infeasible for our problem because the number of oscillations (roughly, the number of tracks) is on the order of a million.

If the function is known to have period p, another approach is to map the input x into the range $(0, p)$ using $x \bmod p$. Common examples include time-of-day or day-of-year inputs for functions that are daily or yearly periodic [29, 15]. An alternative is to map it to $\sin(2\pi x/p)$ and $\cos(2\pi x/p)$ [20]. (This pair of functions has nice properties; they are both continuous and bounded, and weighted sums are equal to other sinusoids with varying phase.) Either way, this approach requires the period to be precisely known ahead of time, and that the input be exactly periodic. Note that such a period cannot be calculated from manufacturer's specifications.

Neurons in a neural net calculate their output by taking a weighted sum of the inputs and applying an *activation function* or *transfer function*, typically $\tanh(x)$ or $\frac{1}{1+e^{-x}}$. Rather than using one of these as the activation function, one may use $\sin(x)$. Unfortunately, $\sin(x)$ does not approach a limit for large x, while $\tanh(x)$ and $\frac{1}{1+e^{-x}}$ do. Lack of a such a limit can lead to instability [52]. Periodic activation functions also introduce many local minima [46].

A powerful tool for time series data is recurrent or delay networks [39, 26]. These feed the network back into itself, so that the network predicts y values from other y values, rather than from x values. This is usually applied to problems with one dimension of recursion, but ours has two, one for the previously accessed sector and one for the current sector to access. With dense data, only one level of recursion is necessary, because the other y values are already known. If the data is sparse, missing values must be recursively computed. Our sampling is (by necessity) so sparse that the recursion would be extremely deep, making computation time impractical.

Park *et al.* used what they call spectral basis neural networks for traffic prediction [42]. These are neural nets with the input augmented with sines and cosines of the raw input. Although similar to our work, they differ in two ways. First, they use every integer frequency in a given range, rather than frequencies that are powers of two. This limits the frequency range they are sensitive to with a given number of sines and cosines. Second, their motivation differs. They do not appear to be interested in finding periodic behavior, but instead use the additional inputs to make the problem linearly separable.

2.3 Fourier analysis

When a signal is sampled at every point on a regular grid, the Fast Fourier Transform (FFT) can be used to compute the Fourier transform in $O(n \log n)$ time, where n is the number of grid points. If the sampling is sparse, however, FFT cannot be used, and a brute force calculation would require $O(kn)$ time, where k is the number of samples. A much more desirable running time would be at most a logarithmic factor away from $O(k)$, which is where sparse Fourier algorithms come in.

Sparse Fourier algorithms assume that the number of non-negligible frequency components is small. Motivations vary, but the most common are to reduce the number of required samples or to reduce the computation time [21]. We do require a sparse sampling, but our bottleneck is primarily computation time rather than number of samples. With that in mind, we looked for an algorithm that is fast to compute. Some can run as fast as $O(k \log k)$ [21], but they make assumptions which do not hold for our problem. Specifically, we want an algorithm which supports non-integer frequencies and multidimensional signals.

A factor that turned out to be important is the existence of non-integer frequencies in our signal (fig. 4). Many sparse Fourier algorithms assume that the frequencies are integer and will result in significant leakage if this assumption is violated [6]. Duarte and Baraniuk describe *spectral compressive sensing* which allows for off-grid frequencies, but it is one-dimensional, and they do not describe runtime bounds [14]. Tang *et al.* have similar results [48].

We would also prefer an algorithm which supports multidimensional Fourier analysis, since our problem is inherently two-dimensional. Unfortunately, the vast majority of sparse Fourier research has been on the one-dimensional case. Chi and Chen build on Tang *et al.*'s results and demonstrate an algorithm for sparse Fourier analysis of a two-dimensional signal with off-grid frequencies [10], but they are unable to provide run time guarantees.

3. FINDING PERIODICITY

We showed previously that recognition of periodicity is crucial for predicting access times [11]. However, the question of how to determine the periodicity is still open. The obvious solution is to use the Fourier transform, which we describe in section 3.1. The Fourier transform turns out to be difficult to calculate for our data set, as well as having inherent limitations. Because of this, we decided to bypass Fourier analysis entirely and let the neural net find the periodicity itself. This solution is described in section 3.2.

From here on out, we define *zone* to mean a contiguous range of logical sectors that lie in tracks of the same length. Tracks which have the same length and are physically adjacent but not logically adjacent are not necessarily part of the same zone.

3.1 Hierarchical Fourier analysis

Considering pairs of sectors (a start sector and an end sector), each of which can take roughly a billion unique values for modern drives, the grid of all 2D periods to search contains roughly 10^{18} points. Computing the Fourier transform over such a large domain is prohibitively expensive. The Fast Fourier Transform (FFT) requires a dense data set, which at 4 bytes per point would take up about 3.5 exabytes,

and would take about 500 million years to capture. Even subsampling 1 out of every 1000 sectors, which is roughly the Nyquist rate given normal track lengths, would yield 3.6 terabytes, and would take about 500 years to capture.

To simplify the problem, we previously made the assumption that interesting frequencies lie precisely on the diagonal of the 2D Fourier transform [11]. With this assumption, the grid size is reduced to 10^9 points. This improves the brute force search from impossible to merely impractical, but this still is not good enough, as it would take about six months to capture the data. To reduce the sampling time, we are forced to rely on a sparse sampling.

Designs requiring Fourier analysis also require boundaries within which that analysis is valid, which for our problem correspond to zone boundaries, or places where the track length changes. Since the boundaries for our problem are not known in advance, we must search for them. A straightforward way to search for these boundaries is to use a divide-and-conquer approach. Fourier analysis is performed recursively on smaller and smaller subregions until consecutive spectra are identical. This is similar to a binary search, except that both branches may be taken.

For example, say that the search range is (0, 1), and there is one boundary at $\frac{1}{4}$. The left and right halves of (0, 1) will have spectra computed. These spectra will not match, because the spectrum for $(\frac{1}{2}, 1)$ will have frequencies that do not show up in $(0, \frac{1}{2})$. Therefore, $(0, \frac{1}{2})$ and $(\frac{1}{2}, 1)$ will each have their left and right halves analyzed. Since the two halves of $(\frac{1}{2}, 1)$ have identical spectra, the recursion for it stops. However, the spectra for $(0, \frac{1}{4})$ and $(\frac{1}{4}, \frac{1}{2})$ do not match, so they are recursively analyzed. At the end of this process, the leaves of the tree are candidate zones. Adjacent regions with identical spectra are merged, leaving $(0, \frac{1}{4})$ and $(\frac{1}{4}, 1)$.

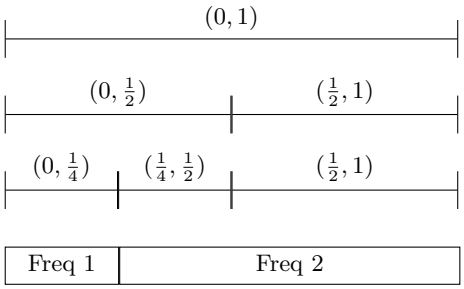


Figure 2: Recursive Fourier analysis can be used to find boundaries where frequencies change.

The need to compare spectra for equality adds additional computation. Because of the large number of frequencies, even strong signals can be swamped by noise when two spectra are compared. We reduce the noise by sparsifying the Fourier transform: a penalty term is added to inhibit noise, and a few rounds of stochastic gradient descent are run.

Unfortunately, this procedure is computationally expensive. We ran a test using a data set of 320,000 reads from the first 1,901,048 sectors of the drive, which covers roughly 8 zones. The analysis was run on a 12-node cluster with 2 dual-core AMD Opteron 2212 processors per node, or 48 cores total. The analysis took 5 hours and 9 minutes. The algorithm has a (probably loose) lower bound linear in the number of sectors, since that determines the number of frequencies to search through in a single Fourier transform computation. Using that lower bound and scaling up to the entire drive (which has 976,773,168 sectors), it would take over 110 days to analyze the entire drive, assuming the same amount of data. Since more data would actually be necessary to maintain the same fidelity over a larger range, the result would be even worse.

3.2 Constructing periods

As detailed in the previous subsection, the Fourier transform is difficult to calculate for our problem. Furthermore, because the data contains many large discontinuities, the Fourier transform is not well-behaved. In particular, a dominant pattern in the data is a sawtooth, whose Fourier transform has harmonics whose coefficients decay as $1/n$. In other words, a very large number of harmonics have significant power. This masks even the fundamental frequency of other waves with lower power.

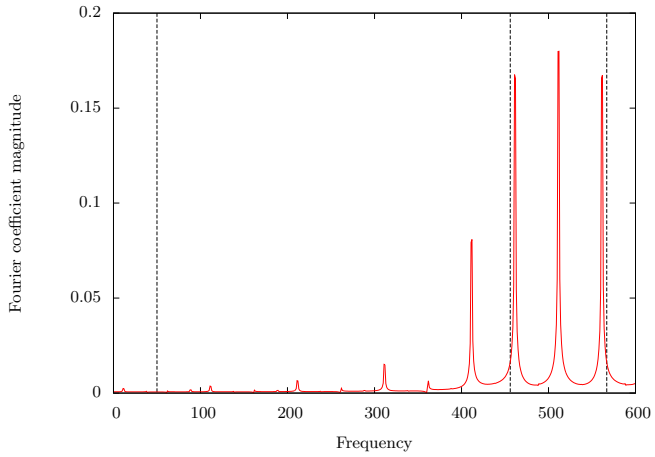


Figure 3: Fourier transform of signal has "interesting" frequencies at the marked locations, but correlation between "interestingness" and Fourier coefficient magnitude is weak.

Signals which are not simple and precisely periodic may also have "interesting" frequencies that do not show up in the Fourier transform. For example, consider:

$$f(x) = \begin{cases} sin(456 \cdot 2\pi x + \phi_k) & sin(50 \cdot 2\pi x) < 0 \\ sin(567 \cdot 2\pi x + \phi_k) & \text{otherwise} \end{cases} \quad (1)$$

where

$$k = \lfloor 2 \cdot 50x \rfloor \quad (2)$$

with ϕ defined to make f continuous. In other words, the frequency of this signal switches between 456 and 567, with the switching occurring with frequency 50. The Fourier transform of this function is seen in fig. 3. None of the frequencies mentioned correspond exactly to a peak in the Fourier transform. Of course, the Fourier transforms of individual zones of this function have sharp peaks at the appropriate frequencies, but in the real data the locations of the boundaries and even the number of zones are not known ahead of time.

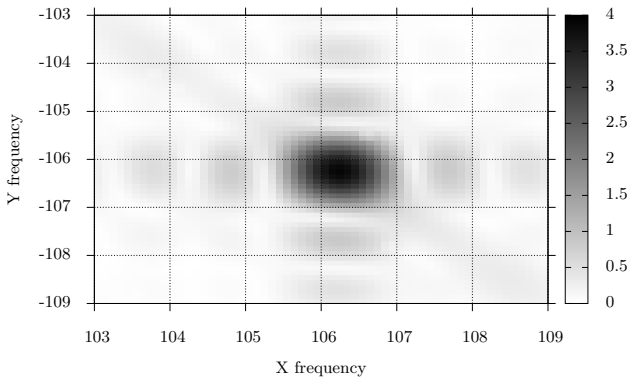


Figure 4: Section of the Fourier transform of access times from the first few tracks, with darker colors indicating higher magnitude. Note that the peak is not centered on a pair of integer coordinates.

Another problem is that our data contains *off-grid* frequencies (fig. 4), or frequencies which are not exact integers. Many sparse Fourier algorithms do not work well with off-grid frequencies [6], and will instead report many frequencies near the frequency of interest, but with lower magnitudes. Finding a sparse Fourier algorithm which is two-dimensional, supports off-grid frequencies, is computationally efficient, is efficient in the number of samples required, has reasonable constant factors, and which generally scales to the size of our problem is nearly impossible.

Because of the difficulties of computation and limitations in the result of the Fourier transform, we decided that the best approach was to let the neural net compute the frequencies for itself. Computing periodic waveforms directly is a difficult problem (see checkerboard problem [47] and two-spirals problem [17, 44]), so we feed the neural net a base set of sinusoids from which it can compute arbitrary sinusoids.

From basic trigonometry, the angle sum formulas for sine and cosine are:

$$\cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta) \quad (3)$$

$$\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta) \quad (4)$$

Adding a time parameter gives:

$$\cos((\alpha+\beta)t) = \cos(\alpha t)\cos(\beta t) - \sin(\alpha t)\sin(\beta t) \quad (5)$$

$$\sin((\alpha+\beta)t) = \sin(\alpha t)\cos(\beta t) + \cos(\alpha t)\sin(\beta t) \quad (6)$$

which means that a neural net can construct sines and cosines with frequency $\alpha+\beta$ very simply given sines and cosines with frequencies α and β. By extension, sinusoids of any integer frequency n can be constructed by a neural net with depth $O(\log\log n)$ given $O(\log n)$ input sinusoids with frequencies of the form 2^k. Of course, this is not necessarily the mechanism by which the neural net computes its output, but it serves as an explanation to show that the idea has theoretical merit.

Since the number of sectors, and therefore the number of periods, is on the order of a billion, and since we use frequencies of the form 2^k, the number of input frequencies is roughly $\log_2 10^9 \approx 30$. Combining an arbitrary subset of these may be too difficult an operation for a neural net with a reasonable number of neurons to perform in a single layer, so we allow the neural net to have multiple hidden layers. This corresponds to the $O(\log\log n)$ depth mentioned in the previous paragraph.

4. INCREMENTAL LEARNING

The problem of learning to compose periods, find the difference between the phases of the composed periods, and generate an appropriate output waveform is too difficult for a neural net to learn all at once. In fact, our attempts to do so failed. One problem is that there is a large number of inputs, about 30 sines and cosines per input request, and learning to combine the appropriate subset of these is a difficult problem in and of itself. Another problem is that different zones of the disk have different track lengths, meaning that different composite periods are required, so the neural net must learn to switch between them.

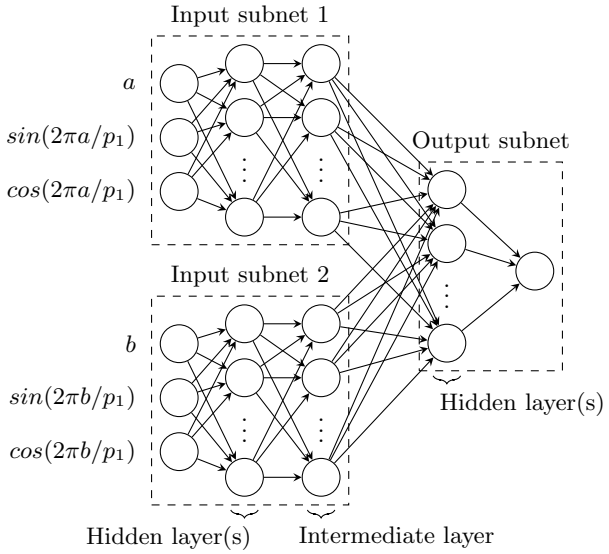


Figure 5: Network architecture. Note that the weights in the two input subnets are identical.

Our intent is that these subproblems are handled by different parts of the neural net, whose architecture is shown in fig. 5. Specifically, period choice and composition should be performed by the input subnets, and the phase comparison and output generation should be performed by the output subnet. By breaking the training into stages that focus on parts of the neural net, the difficulty of each stage of training should be reduced. This idea of multi-stage training shares a conceptual thread with the pretraining used with deep learning for classifying images [16]. In these approaches, the researchers pretrain layers close to the input before moving to the main training phase. Although their motivation is different (making use of unlabeled data), it has much the same effect.

4.1 Approach 1: expanding coverage

To break the training into stages, we start by training with data from a tiny portion of the drive. We then increase the covered portion of the drive, training along the way, until the entire drive is covered. When the covered portion is much less than one composite period, the phase is approximately linear in the sector number. This means that approximating the composite period is very easy, and most of the training is focused on the output subnet, which computes the phase difference and output waveform. As more of the drive is covered, the difficulty of approximating the composite periods is gradually increased.

This approach appears to partially work. The coverage can be expanded from a few sectors to more than one zone, and the error shows that the model is making better than chance predictions. Unfortunately, when we stop expanding the coverage and focus on reducing the error, we are unable to reduce the error to a reasonable amount. We believe this is because the neural net gets stuck in a configuration that is good for the early parts of training, essentially in a local minimum and unable to change to a configuration that is good for the larger area.

4.2 Approach 2: half-zero sampling

An alternative approach is to train the period composition first, and then train the phase differencing. This is accomplished by capturing a training set where the start sector is always sector 0, and the end sector varies. Since one of the sectors is constant, computing the difference is trivial, and training is focused on learning to generate the composite periods. The start sector can then be allowed to vary more and more until it can be any sector on the drive.

We use a workload that consists of random reads, except with half of the reads randomly changed to sector 0. This means that (previous, current) sector pairs in the workload consist of a mix of $(0,0)$, $(0,x)$, $(x,0)$, and (x,y). The $(0,0)$ pairs are discarded, since they are useless, leaving an even mix of $(0,x)$, $(x,0)$, and (x,y). Training on these all at once avoids the local minimum problem in the previous approach, which seems to be caused by changing characteristics of the training set over time. Doing so also avoids the question of when to switch data sets. Although the data is provided all at once, the neural net presumably learns the $(0,x)$ and $(x,0)$ pairs first, since they are easier, and then learns the (x,y) pairs.

5. DISCONTINUITY REMOVAL

A basic characteristic of neural nets is that they assume that the function being approximated is continuous, and so they do not handle discontinuities well. This is very unfortunate, because the function we are approximating is full of large discontinuities. Thankfully, these discontinuities have a structure that makes them removable.

The discontinuities are caused by the rotational nature of a hard disk drive. In this paragraph, seek time is defined to include settle time and head switch time. If the seek time is less than the rotation time, then the head simply waits on the destination track until the destination sector rotates underneath it. If the seek time is greater than the rotation time, the head will miss the sector and must wait for an additional rotation. The discontinuities in the access time function are (mostly) places where the seek time is almost exactly equal to the rotation time, and a slight change in the sector number, which implies a slight change in rotation time, causes a rotation miss. This explanation predicts that the discontinuities will have a height which is equal to the rotational period of the drive, which is borne out by data.

Consider a continuous section of the latency function between two discontinuities. This section runs from a minimum to a maximum (or from a maximum to a minimum). The section can be translated and scaled so that it runs from $-\pi$ to π. If it is now treated as an angle, instead of as a real number, then $-\pi$ and π are equal, so the discontinuity is gone. This is shown geometrically in fig. 6.

In more concrete terms, consider the sawtooth $f(x)$ which has a minimum of $-h$ and a maximum of h. Instead of predicting $f(x)$ directly, we predict $c = \cos\left(\frac{\pi}{h} f(x)\right)$ and $s = \sin\left(\frac{\pi}{h} f(x)\right)$. These are continuous functions of x, so the neural net sees no discontinuities. The value of $f(x)$ can be reconstructed using $\hat{f}(x) = \frac{h}{\pi}\text{atan2}(c,s)$.

The problem is slightly complicated by the fact that while the latency function looks locally like the sawtooth described, globally it has a greater range between its minimum and maximum. To account for this, we use a sliding window approach. The neural net predicts a lower bound for the latency function, l, in addition to c and s. The final predicted latency is then $\hat{f}(x) = \frac{h}{\pi}\text{atan2}(c,s) + 2kh$, where k is the smallest integer such that $\frac{h}{\pi}\text{atan2}(c,s) + 2kh > l$.

6. RESULTS

We tested our approach against two hard disk drives. The first, which we call HD1, is the same as used in our previous paper. It is a Western Digital Black WD5002AALX 500 GB 7200 RPM, with a total of 976,773,168 sectors. The second hard drive, HD2, is a Seagate ST31000340SV 1 TB 7200 RPM, with a total of 1,953,525,168 sectors. Native command queueing is disabled to prevent the drive from reordering requests and biasing toward lower latencies.

We used a genetic algorithm (GA) to find good neural network topologies. The GA chose how many layers to use, how many neurons should be in each layer, which periods to keep, and how the initial weights should be randomized. We run the GA separately for each range and hard disk drive. This may not be necessary in every case. For example, different devices from the same model line will almost certainly be modeled well with the same configuration.

The neural nets were trained on data sets captured with special sampling methods, but testing is always performed with a data set of random reads chosen uniformly over the region of interest.

Predicted versus actual latencies for a neural net predicting latency directly for the first 237,631 sectors of HD1 are shown in fig. 7a. The neural net has 5 hidden layers in the input subnets with sizes 39, 56, 164, 9, and 5, an intermediate layer with size 18, and 2 hidden layers in the output subnet with sizes 51 and 15. The weights were initialized with normally distributed random values with standard deviations

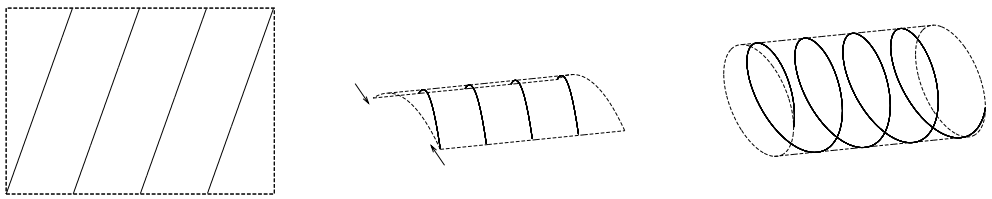


Figure 6: A 2D sawtooth can be rolled up into a 3D helix to remove its discontinuities.

of 0.066, 0.775, 0.872, 0.051, 0.320, 0.355, 0.637, 0.200, and 1.280. The input included sines and cosines with periods of 2^4, 2^8, 2^{10}, 2^{11}, 2^{12}, 2^{14}, 2^{15}, 2^{17}, 2^{18}, and 2^{25}. The mean absolute error is 0.333. If we correct for predictions which are off by exactly one rotation, the mean absolute error falls to 0.176.

Note that at low and high access times, the errors are large. This is caused by the discontinuities in the access time function. When the value to predict lies very close to a discontinuity, the neural net has difficulty predicting which side of the discontinuity the value lies on, and therefore whether it should be large or small. Since the largest and smallest values are on either side of discontinuities, this means that our largest errors are for the largest and smallest access times.

Predicted versus actual latencies for a neural net with the discontinuities removed predicting latency for the first 237,631 sectors HD1 are shown in fig. 7b. The neural net has 5 hidden layers in the input subnets with sizes 55, 109, 63, 29, and 5, an intermediate layer with size 9, and 1 hidden layer in the output subnet with size 49. The weights were initialized with normally distributed random values with standard deviations of 0.649, 0.600, 0.285, 0.271, 0.251, 0.457, 0.268, and 0.102. The input included sines and cosines with periods of 2^3, 2^7, 2^9, 2^{10}, 2^{11}, 2^{13}, 2^{14}, 2^{15}, 2^{16}, 2^{17}, and 2^{18}. The mean absolute error is 0.187. If we correct for predictions which are off by exactly one rotation, the mean absolute error falls to 0.029.

In fig. 8, predicted versus actual latencies are plotted for the first 2^{20} = 1,048,576 sectors of HD1. This covers roughly four zones, meaning that the periodicity changes three times. The plot shows that the neural net is able to handle changes in periodicity without the locations of the changes being determined beforehand. If a method based on Fourier analysis were used, the boundaries would have to be located and the input to the neural net preprocessed. The neural net has 3 hidden layers in the input subnets with sizes 57, 105, and 55, an intermediate layer with size 36, and 3 hidden layer in the output subnet with sizes 11, 21, and 41. The weights were initialized with normally distributed random values with standard deviations of 0.551, 0.929, 0.372, 0.471, 0.049, 0.460, 0.317, and 0.197. The input included sines and cosines with periods of 2^2, 2^{12}, 2^{14}, 2^{15}, 2^{16}, 2^{17}, 2^{18}, and 2^{19}. The mean absolute error is 0.639. If we correct for predictions which are off by exactly one rotation, the mean absolute error falls to 0.061.

In fig. 9, predicted versus actual latencies are plotted for the first 2^{20} = 1,048,576 sectors of HD2. The neural net has 5

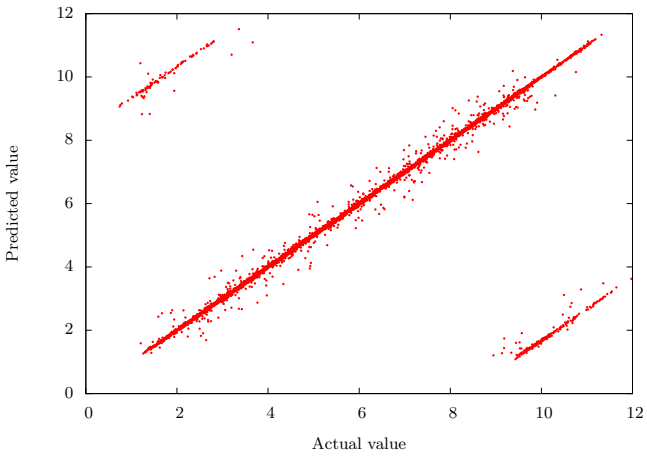
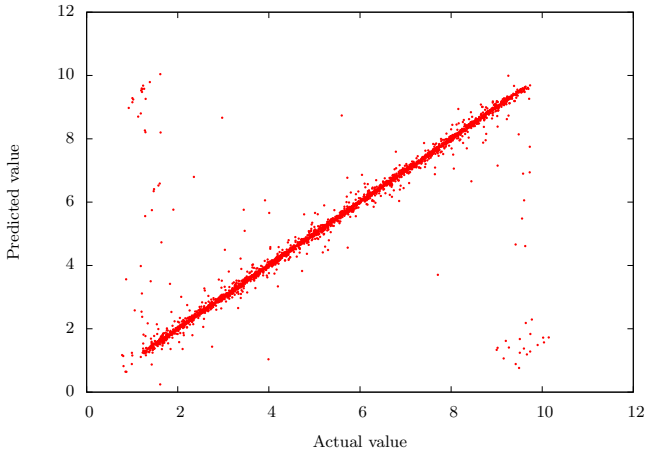


Figure 8: Predicted versus actual latency over the first 2^{20} = 1,048,576 sectors of HD1 with discontinuities removed. The mean absolute error is 0.639. If we correct for predictions which are off by exactly one rotation, the mean absolute error falls to 0.061. For details of the neural net configuration, see section 6.

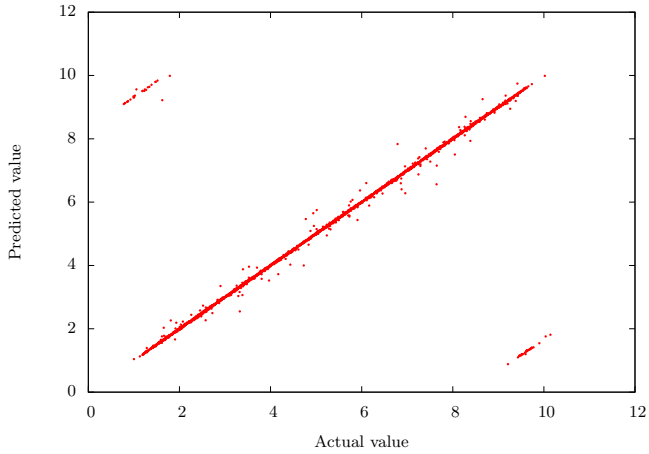
hidden layers in the input subnets with sizes 49, 92, 55, 50, and 10, an intermediate layer with size 8, and 1 hidden layer in the output subnet with size 28. The weights were initialized with normally distributed random values with standard deviations of 0.159, 0.940, 0.219, 0.490, 0.602, 0.612, 0.194, 0.185. The input included sines and cosines with periods of 2^4, 2^{11}, 2^{15}, 2^{16}, 2^{17}, 2^{18}, 2^{19}, 2^{20}, 2^{21}, 2^{24}, 2^{25}, 2^{26}, and 2^{28}. The mean absolute error is 0.202. If we correct for predictions which are off by exactly one rotation, the mean absolute error falls to 0.066.

When tasked with predicting the first 2^{22} = 4,194,304 sectors of HD1, the neural net has 5 hidden layers in the input subnets with sizes 46, 127, 65, 30, and 14, an intermediate layer with size 14, and 1 hidden layer in the output subnet with size 37. The weights were initialized with normally distributed random values with standard deviations of 0.156, 0.594, 0.198, 0.200, 0.232, 0.341, 0.217, and 0.209. The input included sines and cosines with periods of 2^{11}, 2^{14}, 2^{15}, 2^{16}, 2^{17}, 2^{18}, 2^{19}, 2^{20}, 2^{21}, 2^{22}, and 2^{23}. The mean absolute error is 1.048. If we correct for predictions which are off by exactly one rotation, the mean absolute error falls to 0.135.

When tasked with predicting the first 2^{22} = 4,194,304 sectors of HD2, the neural net has 4 hidden layers in the input sub-



(a) Direct latency prediction. The mean absolute error is 0.333. If we correct for predictions which are off by exactly one rotation, the mean absolute error falls to 0.176.



(b) Predictions with the discontinuities removed. The mean absolute error is 0.187. If we correct for predictions which are off by exactly one rotation, the mean absolute error falls to 0.029.

Figure 7: Predicted versus actual latency over the first 237,631 sectors of HD1, without and with discontinuities removed. For details of the neural net configuration, see section 6.

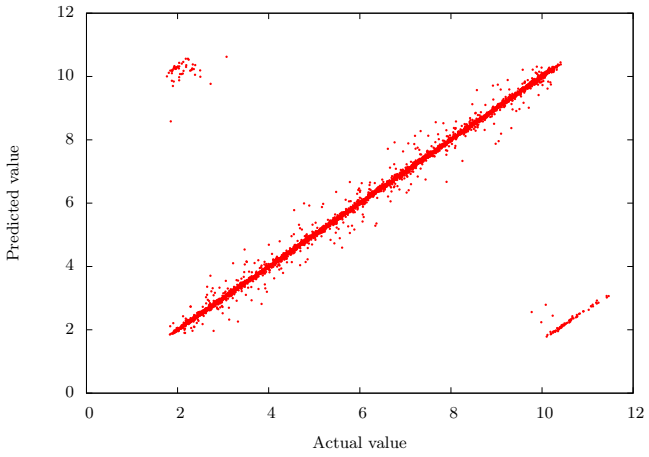


Figure 9: Predicted versus actual latency over the first $2^{20} =$ 1,048,576 sectors of HD2 with discontinuities removed. The mean absolute error is 0.202. If we correct for predictions which are off by exactly one rotation, the mean absolute error falls to 0.066. For details of the neural net configuration, see section 6.

Device	Range size	Requests	MAE	MAE corrected
HD1	237,631	40,000	0.187	0.029
HD1	1,048,576	80,000	0.639	0.061
HD1	4,194,304	160,000	1.048	0.135
HD2	1,048,576	80,000	0.202	0.066
HD2	4,194,304	160,000	0.829	0.170

Table 1: Mean Absolute Error (MAE) in milliseconds for various configurations. "MAE corrected" refers to the error when an integer number of rotational latencies is added to the prediction to move it to the correct period.

nets with sizes 44, 170, 52, and 4, an intermediate layer with size 10, and 1 hidden layer in the output subnet with size 45. The weights were initialized with normally distributed random values with standard deviations of 0.148, 1.763, 0.126, 0.550, 0.883, 0.293, and 0.125. The input included sines and cosines with periods of 2^{11}, 2^{15}, 2^{16}, 2^{17}, 2^{18}, 2^{19}, 2^{20}, 2^{21}, 2^{22}, 2^{23}, and 2^{27}. The mean absolute error is 0.829. If we correct for predictions which are off by exactly one rotation, the mean absolute error falls to 0.170.

Errors for various configurations are shown in table 1. Training was attempted with a data set size of 40,000. If the GA failed to converge, the data set size was doubled until the GA did converge. It is not yet clear how the required data set size scales with the range of sectors covered, but larger ranges cover more varied topology, so it is unsurprising that they require more data.

7. FUTURE WORK

The immediate next step is to expand coverage to the entire hard disk drive. This would include determining how the required data set size scales with the range size.

A relatively simple improvement would be to allow the request size to vary. Accounting for scheduling and queueing effects could be accomplished by providing the past several

requests instead of just one, although this is complicated by the fact that a request's latency can be affected by the requests issued *after* it if requests can be responded to out of order. Even more difficult still would be allowing writes and non-random reads, as the performance of both of these both relies on cache behavior. Not only is cache behavior potentially complex, it introduces long-lived state.

8. CONCLUSION

Although neural nets are poorly suited for directly approximating a high-frequency periodic function, they can provide good results when given high-frequency sines and cosines as additional inputs. When providing these sines and cosines, neural nets can be trained to predict hard disk drive access times for random reads, even across zones of the drive with varying periodicity.

Results can further be improved by removing the discontinuities in the latency function. This requires knowing the rotational latency of the hard disk drive, but it may be a small price to pay for increased precision.

Because the error for individual requests is low, unless the latency is off by an integer number of rotations, rotation misses (or mispredictions thereof) are by far the largest source of error. If these are random, caused by fluctuations inside the device, then the prediction error is nearly as low as it can be. If the misses are not random, then predicting them accurately should be the next focus of research.

9. REFERENCES

[1] Quantum atlas 10k ii. `http://www.seagate.com/staticfiles/maxtor/en_us/documentation/data_sheets/atlas_10k_ii_datasheet.pdf`, 2000. AIIDS0600.

[2] Seagate breaks capacity ceiling with world's first 3 terabyte external desktop drive. `http://media.seagate.com/2010/06/seagatetechnology/seagate-breaks-capacity-ceiling-with-worlds-first-3-terabyte-external-desktop-drive/`, June 2010.

[3] WD caviar green series disti spec sheet. `http://www.wdc.com/wdproducts/library/SpecSheet/ENG/2879-701229.pdf`, January 2012. 2879-701229-A25.

[4] Eric Anderson, Susan Spence, Ram Swaminathan, Mahesh Kallahalla, and Qian Wang. Quickly finding near-optimal storage designs. *ACM Transactions on Computer Systems (TOCS)*, 23(4):337–374, 2005.

[5] D. Ashby, B. Norman, and K. Tan. *Barracuda 4LP Family - Product Manual*. Seagate, D edition, Feb 1998.

[6] Petros Boufounos, Volkan Cevher, Anna C Gilbert, Yi Li, and Martin J Strauss. What's the frequency, kenneth?: Sublinear fourier sampling off the grid. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 61–72. Springer, 2012.

[7] Cristian Bucilă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM, 2006.

[8] John S. Bucy, Jiri Schindler, Steven W. Schlosser, Gregory R. Ganger, and Contributors. *The DiskSim Simulation Environment Version 4.0 Reference Manual*. Carnegie Mellon University, Pittsburgh, PA, May 2008.

[9] Ying Chen, Windsor W. Hsu, and Honesty C. Young. Logging RAID - an approach to fast, reliable, and low-cost disk arrays. In Arndt Bode, Thomas Ludwig, Wolfgang Karl, and Roland Wismüller, editors, *Euro-Par 2000 Parallel Processing*, volume 1900 of *Lecture Notes in Computer Science*, pages 1302–1311. Springer Berlin Heidelberg, 2000.

[10] Yuejie Chi and Yuxin Chen. Compressive recovery of 2-d off-grid frequencies. In *Signals, Systems and Computers, 2013 Asilomar Conference on*, pages 687–691. IEEE, 2013.

[11] Adam Crume, Carlos Maltzahn, Lee Ward, Thomas Kroeger, and Matthew Curry. Automatic generation of behavioral hard disk drive access time models. In *Mass Storage Systems and Technologies, 2014. MSST 2014. 30th IEEE Conference on*. IEEE, 2014.

[12] Chenjun Dai, Guiquan Liu, Lei Zhang, and Enhong Chen. Storage device performance prediction with hybrid regression models. In *PDCAT'12*, Beijing, China, December 2012.

[13] Jesus Dìaz. This is the largest SATA drive in the world. `http://gizmodo.com/5667594/this-is-the-largest-sata-drive-in-the-world`, October 2010.

[14] Marco F Duarte and Richard G Baraniuk. Spectral compressive sensing. *Applied and Computational Harmonic Analysis*, 35(1):111–129, 2013.

[15] David Elizondo, Gerrit Hoogenboom, and RW McClendon. Development of a neural network model to predict daily solar radiation. *Agricultural and Forest Meteorology*, 71(1):115–132, 1994.

[16] Dumitru Erhan, Pierre-Antoine Manzagol, Yoshua Bengio, Samy Bengio, and Pascal Vincent. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *International Conference on artificial intelligence and statistics*, pages 153–160, 2009.

[17] Scott E Fahlman and Christian Lebiere. The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, February 1990.

[18] Silvia Ferrari and Robert F Stengel. Smooth function approximation using neural networks. *Neural Networks, IEEE Transactions on*, 16(1):24–38, 2005.

[19] J.D. Garcia, L. Prada, J. Fernandez, A. Nunez, and J. Carretero. Using black-box modeling techniques for modern disk drives service time simulation. In *Simulation Symposium, 2008. ANSS 2008. 41st Annual*, pages 139 –145, april 2008.

[20] M.W. Gardner and S.R. Dorling. Neural network modelling and prediction of hourly NO_x and NO_2 concentrations in urban air in london. *Atmospheric Environment*, 33(5):709 – 719, 1999.

[21] A Gilbert, Piotr Indyk, Mark Iwen, and Ludwig Schmidt. Recent developments in the sparse fourier transform: A compressed fourier transform for big data. *Signal Processing Magazine, IEEE*, 31(5):91–100, 2014.

[22] Jongmin Gim and Youjip Won. Extract and infer quickly: Obtaining sector geometry of modern hard disk drives. *Trans. Storage*, 6:6:1–6:26, July 2010.

[23] Allon Guez and Ziauddin Ahmad. Solution to the inverse kinematics problem in robotics by neural networks. In *Neural Networks, 1988., IEEE International Conference on*, pages 617–624. IEEE, 1988.

[24] Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the Marquardt algorithm. *Neural Networks, IEEE Transactions on*, 5(6):989–993, 1994.

[25] Hai Huang, Wanda Hung, and Kang G Shin. FS2: dynamic data replication in free disk space for improving disk performance and energy consumption. *ACM SIGOPS Operating Systems Review*, 39(5):263–276, 2005.

[26] Sarangapani Jagannathan and Frank L Lewis. Multilayer discrete-time neural-net controller with guaranteed performance. *Neural Networks, IEEE Transactions on*, 7(1):107–130, 1996.

[27] Terence Kelly, Ira Cohen, Moises Goldszmidt, and Kimberly Keeton. Inducing models of black-box storage arrays. Technical Report HPL-2004-108, HP Laboratories, Palo Alto, CA, June 2004.

[28] K Kosanovich, A Gurumoorthy, E Sinzinger, and M Piovoso. Improving the extrapolation capability of neural networks. In *Intelligent Control, 1996., Proceedings of the 1996 IEEE International Symposium on*, pages 390–395. IEEE, 1996.

[29] Jaimyoung Kwon, Benjamin Coifman, and Peter Bickel. Day-to-day travel-time trends and travel-time prediction from loop-detector data. *Transportation Research Record: Journal of the Transportation Research Board*, 1717(1):120–129, 2000.

[30] Abigail S. Lebrecht, Nicholas J. Dingle, and William J. Knottenbelt. A performance model of zoned disk drives with I/O request reordering. In *Proceedings of the 2009 Sixth International Conference on the Quantitative Evaluation of Systems*, QEST '09, pages 97–106, Washington, DC, USA, 2009. IEEE Computer Society.

[31] Mingqiang Li and Jiwu Shu. DACO: A high-performance disk architecture designed specially for large-scale erasure-coded storage systems. *Computers, IEEE Transactions on*, 59(10):1350–1362, 2010.

[32] D.J. Lingenfelter, A. Khurshudov, and D.M. Vlassarev. Efficient disk drive performance model for realistic workloads. *Magnetics, IEEE Transactions on*, 50(5):1–9, May 2014.

[33] Liu Liu, Zhen Han Liu, Lu Xu, and JunWei Zhang. FBctlr - a novel approach for storage performance virtualization. In *Computational Intelligence and Security (CIS), 2011 Seventh International Conference on*, pages 268–272. IEEE, 2011.

[34] Liu Liu, Lu Xu, Zhen Han Liu, and JunWei Zhang. QClock: An interposed scheduling algorithm for performance virtualization in shared storage systems. In *Networked Computing (INC), 2011 The 7th International Conference on*, pages 17–21. IEEE, 2011.

[35] Ning Liu, Jason Cope, Philip Carns, Christopher Carothers, Robert Ross, Gary Grider, Adam Crume, and Carlos Maltzahn. On the role of burst buffers in leadership-class storage systems. In *MSST/SNAPI 2012*, Pacific Grove, CA, April 16 - 20 2012.

[36] Yonggang Liu, Renato Figueiredo, Dulcardo Clavijo, Yiqi Xu, and Ming Zhao. Towards simulation of parallel file system scheduling algorithms with PFSsim. In *Proceedings of the 7th IEEE International Workshop on Storage Network Architectures and Parallel I/O (May 2011)*, 2011.

[37] Michael P. Mesnier, Matthew Wachs, Raja R. Sambasivan, Alice X. Zheng, and Gregory R. Ganger. Modeling the relative fitness of storage. In *SIGMETRICS 2007*, 2007.

[38] L. Newman and J. Nowitzke. *Cheetah 9LP Family - Product Manual*. Seagate, C edition, August 1998.

[39] Gregory Noone and Stephen D Howard. Investigation of periodic time series using neural networks and adaptive error thresholds. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1541–1545. IEEE, 1995.

[40] Alberto Núñez, Javier Fernández, Rosa Filgueira, Félix García, and Jesús Carretero. SIMCAN: A flexible, scalable and expandable simulation platform for modelling and simulating distributed architectures and applications. *Simulation Modelling Practice and Theory*, 20(1):12–32, 2012.

[41] Ron Oldfield. Personal communication, January 2013.

[42] Dongjoo Park, Laurence R Rilett, and Gunhee Han. Spectral basis neural networks for real-time travel time forecasting. *Journal of Transportation Engineering*, 125(6):515–523, 1999.

[43] Bruce E Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3-4):373–384, 1996.

[44] Yi Shang and Benjamin W Wah. Global optimization for neural network training. *Computer*, 29(3):45–54, 1996.

[45] Y Shiroishi, K Fukuda, I Tagawa, H Iwasaki, S Takenoiri, H Tanaka, H Mutoh, and N Yoshikawa. Future options for HDD storage. *Magnetics, IEEE Transactions on*, 45(10):3816–3822, 2009.

[46] Josep M Sopena, Enrique Romero, and Rene Alquezar. Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, volume 1, pages 323–328. IET, 1999.

[47] Donald F. Specht and P.D. Shapiro. Generalization accuracy of probabilistic neural networks compared with backpropagation networks. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume i, pages 887–892 vol.1, 1991.

[48] Gongguo Tang, Badri Narayan Bhaskar, Parikshit Shah, and Benjamin Recht. Compressed sensing off the grid. *Information Theory, IEEE Transactions on*, 59(11):7465–7490, 2013.

[49] Alexander Thomasian. Survey and analysis of disk scheduling methods. *ACM SIGARCH Computer Architecture News*, 39(2):8–25, 2011.

[50] Elizabeth Varki, Allen Hubbe, and Arif Merchant.

Improve prefetch performance by splitting the cache replacement queue. In *Advanced Infocomm Technology*, pages 98–108. Springer, 2013.

[51] Mengzhi Wang, Kinman Au, Anastassia Ailamaki, Anthony Brockwell, Christos Faloutsos, and Gregory R. Ganger. Storage device performance prediction with CART models. In *MASCOTS 2004*, 2004.

[52] K-W Wong, C-S Leung, and S-J Chang. Use of periodic and monotonic activation functions in multilayer feedforward neural networks trained by extended Kalman filter algorithm. In *Vision, Image and Signal Processing, IEE Proceedings-*, volume 149, pages 217–224. IET, 2002.

[53] Tao Xie and Yao Sun. Dynamic data reallocation in hybrid disk arrays. *Parallel and Distributed Systems, IEEE Transactions on*, 21(9):1330–1341, 2010.

[54] Zarita Zainuddin and Ong Pauline. Function approximation using artificial neural networks. *WSEAS Transactions on Mathematics*, 7(6):333–338, 2008.

[55] Lei Zhang, Guiquan Liu, Xuechen Zhang, Song Jiang, and Enhong Chen. Storage device performance prediction with selective bagging classification and regression tree. *Network and Parallel Computing*, pages 121–133, 2010.