Multi-resolution Resource Behavior Queries Using Wavelets

Jason Skicewicz Peter A. Dinda Jennifer M. Schopf
{jskitz, pdinda, jms}@cs.northwestern.edu
Department of Computer Science
Northwestern University

Abstract

Different adaptive applications are interested in the dynamic behavior of a resource over different fine- to coarse-grain time-scales. The resource's sensor runs at some fine-grain resource-appropriate sampling rate, producing a discrete-time resource signal. It can be very inefficient to to answer a coarse-grain application query by directly using the fine-grain resource signal. We address this gap between the sensor and its different client applications with a new query model that explicitly incorporates time-scale as a parameter. The query model is implemented on top of an inherently multi-scale wavelet-based representation of the signal (which could be communicated over a set of multicast channels.) A query uses only the wavelet coefficients necessary for its time-scale (and thus could listen to a subset of the channels), greatly reducing the data that need to be communicated. We present very promising initial results on host load signals, showing the tradeoff between compactness and query error. Finally, we describe some of the other operations that the wavelet representation enables.

1. Introduction

Applications running on shared, unreserved distributed computing environments must adapt to changing resource supply, either through their own machinations or through the services of a scheduler or other adaptation agent [1]. In either case, measurement of resource supply, often viewed through the lens of prediction, serves as the basis for making adaptation decisions. In this paper we address periodic time-series measurements (i.e., discrete-time signals) of resources that are supplied in a stream by some sensor. Examples of systems that can provide or predict such time-series data include Remos [9], the Network Weather Service [15], and RPS [5]. We focus here specifically on host load, a signal with which the running time of tasks strongly correlates, and that has been shown to be quite predictable over the short range (1 to 30 seconds) [3, 6].

A tension exists between sensors and the applications and schedulers that they serve because different applications are interested in the behavior of the signal over different time-scales. For example, a real-time scheduling advisor for an interactive application [4, Chapter 6] may be interested in host load over the past second, while a scheduler for a parallel application may be interested in host load over minutes or hours [14]. Also, while some applications desire periodic updates from a measurement stream, others are interested in aperiodically querying those streams for averages over intervals of time.

Sensor developers have addressed the need for diverse queries by sampling at a high frequency that corresponds to the smallest query interval or else is high enough to capture all the dynamics of the underlying resource. These high frequency samples are then delivered to the application which uses them to compute an average over the query interval. For most queries this results in excessive consumption of network bandwidth. It is possible for the sensor to adjust the frequency of measurement on a per-application basis, but this ties sensors and applications quite closely and can impede scalability.

We believe that this tension can be resolved by representing the measurement streams in the wavelet-domain. Wavelets are a mathematical technique for simultaneously representing a signal's time and frequency (scale) behavior. This representation inherently provides the multi-resolution view needed by applications while still preserving the signal's important dynamics regardless of time-scale or compression level. This paper outlines our approach and provides an initial evaluation of these techniques on an interesting representative signal, host load. Wavelet techniques have been previously applied to characterize and generate network traffic [13, 7]. We are using them to help answer dynamic application queries.

Transforming host load signals into the wavelet-domain is very inexpensive, and both stream and interval queries can be answered rather accurately on the transformed signal despite the extremely high compression made possible by the wavelet representation.

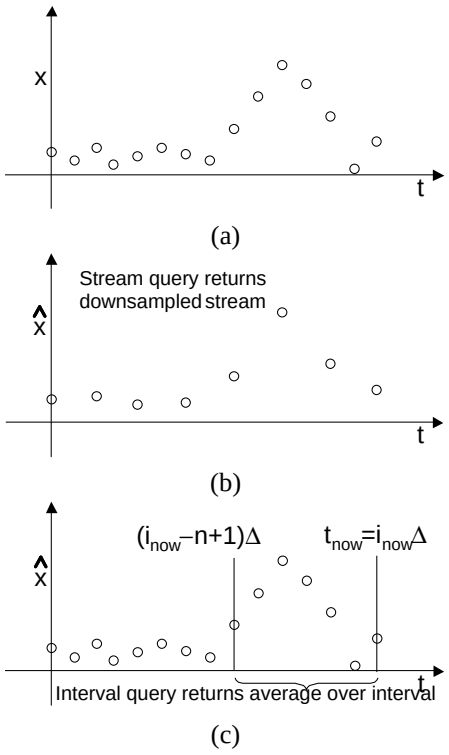


Figure 1. Schematic representation of query model: (a) raw signal, (b) downsampled stream query, (c) interval query.

2. Query model

Distributed applications can use time-series measurements (signals) in different ways. Some are interested in the streams of values themselves, while others want signal averages over intervals of time. The desired sampling rates and the intervals for these queries can also vary widely. Another important observation is that the accuracy that applications require of their queries can vary. Independent of applications, a sensor must attempt to appropriately sample the dynamics of what it is measuring, obeying the Nyquist Criterion [12, Chapter 8].

These broadly varying demands of applications and sensors create a tension that we wish to resolve. To do so, we must clearly define the interaction between applications and sensors, that is, the query model that the sensors support.

Our query model includes both streaming queries and interval queries. These queries operate on the *reconstructed* signal—the client specifies the level of accuracy required, and the reconstruction is done according to the requirements of the query.

Figure 1 illustrates our query model. In (a) we show the discrete-time signal $\langle x_i \rangle$ produced by the sensor. This time-series is assumed to be captured at an appropriate rate $f_s = 1/\Delta$ to capture the dynamics of the underlying signal that is measured. The current time is represented by $t_{now} = i_{now}\Delta$, that is, time is discretized according to the original sample rate of the signal.

In addition to the measured series, we introduce a signal $\langle \hat{x}_i \rangle$ that attempts to sufficiently represent the original signal. This time-series is called the reconstructed signal. The point of using the reconstructed signal is that it may be far less expensive to acquire, communicate, and use than the original signal. Furthermore, the reconstruction allows us to decouple the underlying signal and the use of that data by an application.

There is an error between the reconstructed signal and the original signal, $e_i = \hat{x}_i - x_i$, that forms an error signal $\langle e_i \rangle$. We discuss the use of wavelets to produce the reconstructed signal $\langle \hat{x}_i \rangle$ in the next section. In the limit, the reconstructed signal is identical to the original.

Streaming query: Some applications are interested in periodic updates, perhaps at some rate slower than the raw sampling rate of the signal, as shown in Figure 1(b). For example, a conferencing application may want to modulate the quality of its video streams in step with changing bandwidth on a network path. It would then subscribe to the network sensor at its frame rate, which may well be different from the sampling rate used by the sensor. The application's requested rate (the frame rate in the example) also bounds the maximum frequency in the sensor's output that can be resolved. Lower frequency updates imply lower communication and computation demands, while higher frequency updates imply the opposite.

The form of the streaming query is

$$StreamQuery(f_q) \rightarrow \langle \hat{x}_i \rangle, \hat{\sigma}_e^2,$$

where f_q is the desired measurement rate from the query and is constrained to be $\leq f_s$. The value $\hat{x}_i$ is formed by limiting the reconstructed signal to resolve frequencies up to $f_q/2$ and then sampling at rate f_q. It is as if the original signal were sub-sampled to f_q, except that the reconstructed signal is used, and the result is an estimate. Notice that because it is an estimate, there can be error—the error sequence $\langle e_i \rangle$ may be nonzero. Because of this, the streaming query also returns an estimate of the variance of the error sequence ($\hat{\sigma}_e^2$). It may also return a more detailed report of the error variance, such as a covariance matrix.

Interval query: Other applications are interested in average behavior over intervals of time. In an interval query, the application cares about the full dynamics of the signal, but only as far as it affects the average over a fixed time interval. Figure 1(c) illustrates this model. An example of such

an application is a parallel scheduler that needs to know how many compute cycles were available over the past hour.

Because the reconstructed signal from which the query is made may differ from the original signal, there is also an error associated with this form of query. In our model, the application specifies its tolerance for this error in the form of a confidence level and a frequency representing the rate of the samples that will be averaged.

The form of the interval query is

$$IntervalQuery(N, c, f_q) \rightarrow avg(N), [avg_{low}(N), avg_{high}(N)]$$

where

$$avg(N) = \frac{1}{N} \sum_{i=i_{now}-N+1}^{i_{now}} \hat{x}_i,$$

and $[avg_{low}(N), avg_{high}(N)]$ is the c % confidence interval about $avg(N)$.

Similar to the streaming query, $f_q/2$ is the maximum frequency that must be resolved in the reconstructed signal. Essentially, a lower f_q means that the reconstruction can be cheaper, but the confidence interval will be wider, while a higher f_q means that the reconstruction is more expensive, but the confidence interval is tighter. By adjusting f_q the application can trade off between these two.

3. Wavelet representations

One way to implement the query model presented in the last section is to use a multi-resolution representation of the original measurement stream. This representation must not only produce measurements at a particular rate requested by the application, but must also preserve the time and frequency dynamics of the original measurement stream. The wavelet-domain representation, where the signal is represented by a tree of coefficients that combine time- and frequency-domain information, meets these needs. Time-domain signals are readily converted to the wavelet-domain via the discrete wavelet transformation [16, 11]. Figure 6, which we will elaborate on later, shows a multi-resolution representation of host load computed using wavelets.

The tree-like structure of the wavelet transformation is shown to the left of the network cloud in Figure 2(a). The transform receives measurements, x_n at sample rate f_s, from the sensor and provides a multi-level flow of data, a multi-scale representation of the sensor's output, to the network. More specifically, this structure is known as Mallat's tree algorithm [11, Chapter 17]. Mallat's algorithm calculates the wavelet coefficients in linear time in the size of the input signal, $O(n)$, where n is the length of the input signal. The number of decomposition levels M in the structure

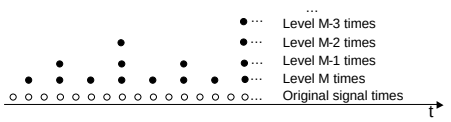


Figure 3. Timing of wavelet coefficients from streaming transform.

is a function of the length of the input signal, and is determined by $n = 2^M$. It is important to note that the transform can also be operated continuously in a streaming manner, in which the number of levels in the wavelet structure increases logarithmically with the number of samples.

The structure is essentially comprised of low-pass filters (LPF) and high-pass filters (HPF) followed by down sampling by 2. At the first stage, the LPF and HPF shown in the figure essentially split the frequency attributes of the original signal in half, yielding two frequency bands, the low ranging from $[0, \frac{f_s}{4}]$, and the high ranging from $[\frac{f_s}{4}, \frac{f_s}{2}]$. The information of the input signal has been split in two and thus the output signals of the filters have twice as much bandwidth information as needed. Because of this redundancy, the output signals can be downsampled while still retaining all of the information. This downsampling operation is repeated at each level.

The Level M decomposition is the output of the high frequency, downsampled measurement stream. To decompose levels further, the output of the low frequency downsampled measurement stream of the same stage is then input into an identical filter-bank, downsample structure. This structure is continued until we have resolved the lowest frequency components. The lowest frequency range of the decomposition, Level 1, represents the coarse grain information contained in the original input signal.

The operation described above is shown pictorially in Figure 2(b), where an input signal $\langle x_i \rangle$ that spans the frequency band $[0, \frac{f_s}{2}]$ is decomposed into a M-level decomposition. If the decomposition only consists of 4 ($M = 4$) levels, then the length of the input signal is only 16 samples long, resulting in a three stage filter bank structure shown in the top portion of Figure 2(a) to the left of the network cloud. The corresponding frequency bands are the first four in Figure 2(b).

As samples stream through the discrete wavelet tree structure, the wavelet coefficients have a unique timing signature, as shown in Figure 3. From this decomposition structure and stream timing, an application can subscribe to the set of levels that best fits the granularity that it requires. For example, the decomposition levels can be sent over the network, each on its own multicast channel.

While the transform is $O(n)$, and the streaming trans-

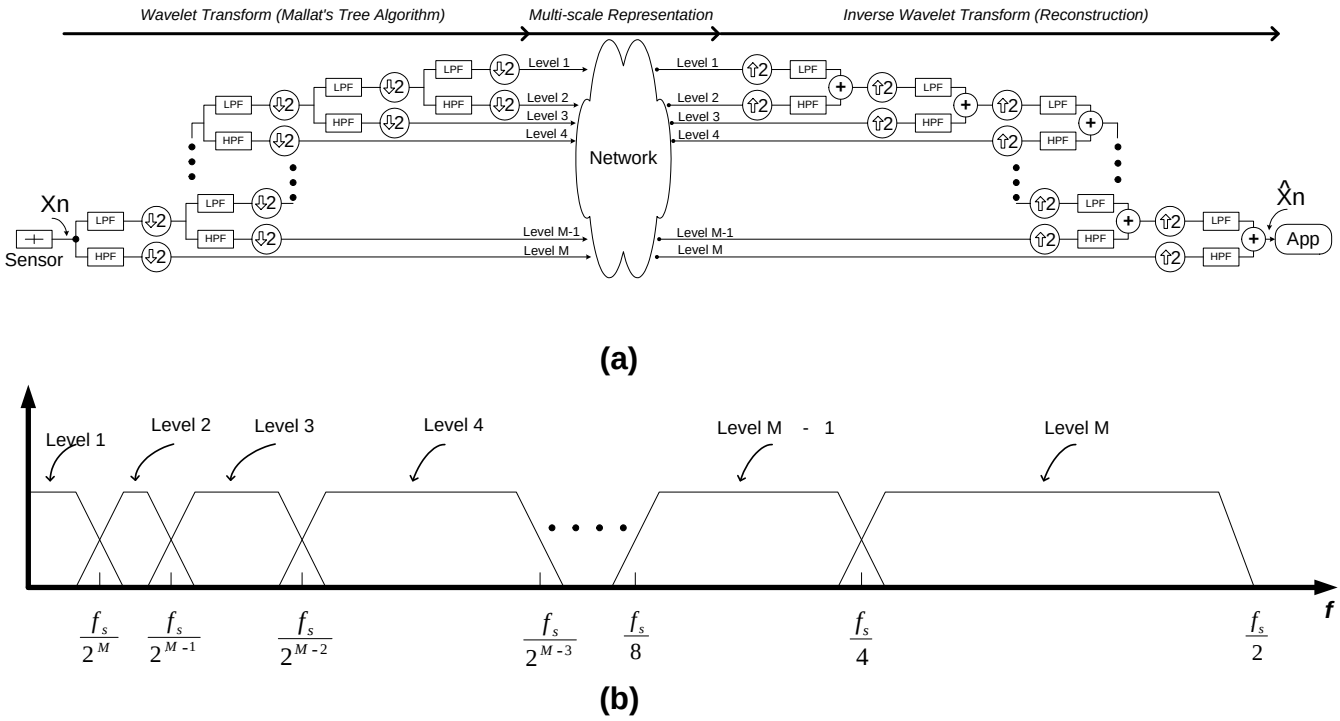


Figure 2. Streaming wavelet system: (a) diagram of the system, (b) frequency representation.

form does at most $O(\lg n)$ work per sample, the actual computational cost of the transform depends on the order of the filters. In our analysis, we use order 8 filters, often called *D8 Wavelets* [2]. The slow interpreted Matlab code we use here takes approximately 1.8 CPU seconds on a 667 MHz Pentium III machine to transform 8192 samples. A streaming version would take less than 220 μs of CPU time per second for 1 Hz load signals (< 0.1 % CPU utilization).

Figure 4(a) shows a host load trace that spans 8192 seconds, approximately 2.5 hours. From this trace, the discrete wavelet transform was applied to the input signal, yielding a set of wavelet coefficients. In Figure 4(b) the mean square energy of each wavelet coefficient is shown, per level of the decomposition. Note that level 0 represents the DC (0 Hz) energy of the input signal (the long term mean). From this picture, the granularity of each level becomes clear, and in addition the time bursts seen in the input signal appear to be located at levels 7 through 10 at various locations along the x-axis. Moving across the x-axis of the plot represents moving in time. This phenomenon becomes more clear by looking at Figure 4(c), a contour map of Figure 4(b).

From the decomposed representation of the input signal, and the query parameters, a level can be chosen such that it contains a sampling rate high enough to satisfy the query. The level is chosen according to the following constraint

$$\frac{f_s}{2^{M-level+2}} \leq f_q \leq \frac{f_s}{2^{M-level+1}},$$

where f_s is the sample rate of the input signal, M is the number of levels in the wavelet decomposition, and $level$ is the level required in order to resolve the desired rate of the query's f_q.

When the level of the decomposition is found according to the above constraint, a reconstruction of the original input signal is performed by acquiring all of the wavelet coefficients from the desired $level$ and the wavelet coefficients from all levels below the chosen one (higher in the tree). This provides the output reconstruction signal with as much information as possible below the desired rate.

In using just the levels necessary for the query's f_q, we are effectively using a query-appropriate compressed representation of the signal which is cheaper to communicate—the application need only listen to *some* of the multicast channels. Figure 5 gives an example of the amount of compression that can be achieved by eliminating levels of the representation and the corresponding impact on the peak frequency that can be resolved.

A diagram of the reconstruction structure, known as the inverse discrete wavelet transform is shown to the right of the network cloud in Figure 2(a). Any application that subscribes to a particular subset of the levels of the decomposition, will require this structure in order to have a representation of the original input signal.

The wavelet transform and inverse wavelet transform pair is actually loss-less—by including all the levels in the

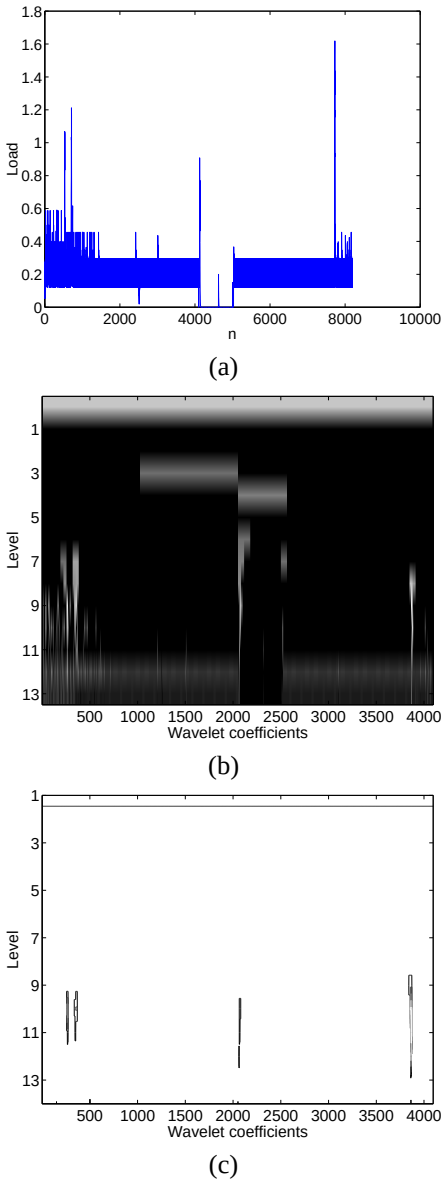


Figure 4. Example wavelet decomposition for host load: (a) host load trace for 8192 seconds, (b) mean square energy of the wavelet coefficients (black indicates low energy, light indicates higher energy), (c) contour of the mean square energy.

reconstruction we end up with the original signal. However, if all levels are not included there is an associated error between the reconstruction and the original signal. By choosing levels as described above, we are basically introducing lossy compression that is appropriate for the query. We will quantify the error that this introduces in Section 4.

Level	Frequency Resolved	Coeffs in level	Coeffs cumulative	Compression (% of total coeffs)
DC	0	1	1	0.01
1	$f_s/8192$	1	2	0.01
2	$f_s/4096$	2	4	0.03
3	$f_s/2048$	4	8	0.09
4	$f_s/1024$	8	16	0.18
5	$f_s/512$	16	32	0.38
6	$f_s/256$	32	64	0.77
7	$f_s/128$	64	128	1.55
8	$f_s/64$	128	256	3.12
9	$f_s/32$	256	512	6.25
10	$f_s/16$	512	1024	12.50
11	$f_s/8$	1024	2048	25.00
12	$f_s/4$	2048	4096	50.00
13	$f_s/2$	4096	8192	100

Figure 5. Wavelet compression gains

Figure 6 shows the reconstruction at different levels of a representative 8192 second segment of a host load trace. As more levels are added in the reconstruction, we capture more and more dynamics of the original signal. Also, when using just the lower, coarse-grained levels, the reconstruction tracks the dynamics of the original signal in a smoother fashion due to the absence of high frequency information.

The benefit of using our wavelet representation instead of a more traditional filtering with downsampling approach is that our approach provides a general, application-independent solution to obtaining measurement streams at different rates. In our approach, a particular measurement rate requested by the application can be directly resolved into a set of levels. A more traditional approach would require the design of per-application low-pass filters with an appropriate downsampling rate in order to satisfy the queries of an application.

Our wavelet representation also enables other techniques that can be applied to the wavelet coefficient stream. One well-known technique that we are currently working with, wavelet de-noising [10], promises to further reduce the number of samples being sent over the network. De-noising throws out coefficients based on their energy, not their level, enabling even further compression gains at a given level of lossy-ness. This may also be a useful technique in lowering storage constraints, enabling interval queries over longer time intervals. We will say more about the techniques we are currently studying in Section 5.

4. Performance evaluation

To evaluate our wavelet-based approach to answering the queries of the previous section, we considered several of Digital Unix 5 second load average traces described in a previous paper [3] as well as a CPU usage trace from a

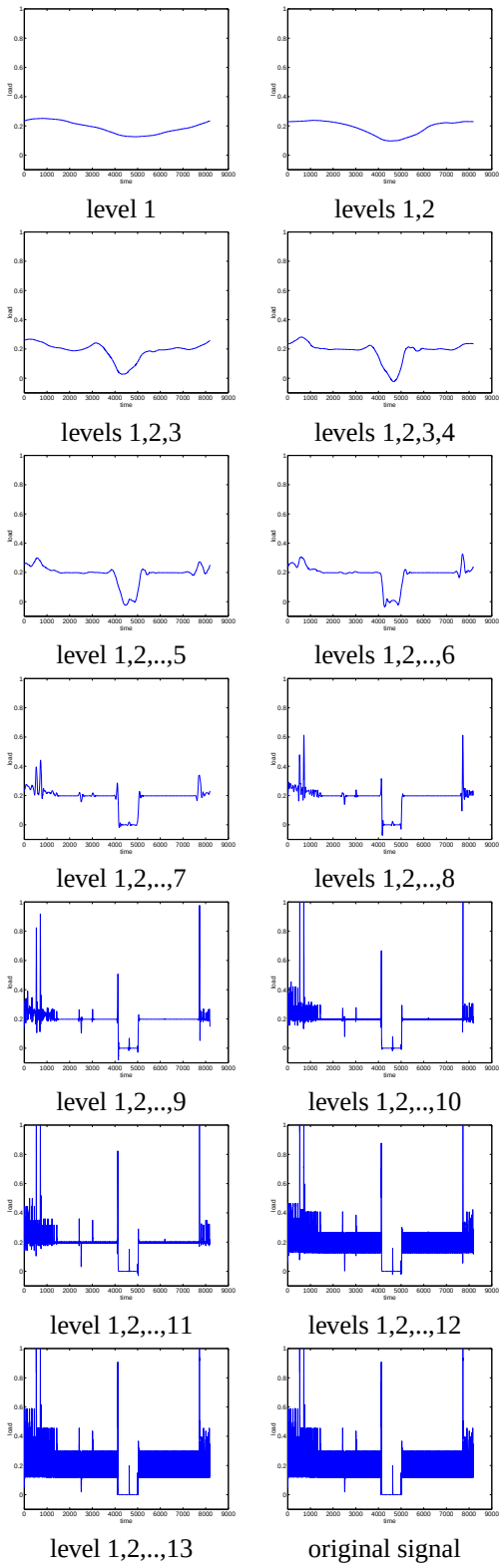


Figure 6. Example of wavelet reconstruction for host load.

Windows 2000 machine collected using WatchTower [8]. WatchTower measures CPU usage as a percentage of the total available CPU. The specific traces are the following:

- axp0: an interactive machine with high load in a production cluster at the Pittsburgh Supercomputing Center (PSC), collected in August, 1997.
- axp7: a batch machine in the PSC's production cluster, collected in August, 1997.
- sahara: a large-memory compute server at Carnegie Mellon University (CMU), collected in August, 1997.
- manchester-2: an interactive machine in a research cluster at CMU, collected in August, 1997.
- themis: a desktop workstation at CMU, collected in August, 1997.
- tlab-03: a teaching lab machine at Northwestern University, collected in May, 2001.

The traces are chosen to span the kinds of machines we have observed in the earlier paper [3], both in terms of nominal classification ("desktop") and in terms of trace statistics. In addition, axp0 represents a much more heavily loaded machine than the others, a distinction that is important because heavily loaded machines tend to be more dynamic. Because most of our results on the other traces are qualitatively similar, we will structure our discussion around the axp7 trace, pointing out where the others differ as needed. The Digital Unix traces are publicly available via http://www.cs.nwu.edu/∼pdinda/LoadTraces. We will make the WatchTower trace available to interested parties.

We implemented mockups of the stream and interval queries around D.E. Newland's Wavelet Toolbox [11]. We segmented the entire 12 day trace axp7 (1 Hz sample rate, over one million samples) into 8192 sample segments to evaluate stream queries. For interval queries, we chose random intervals of different sizes within the segments. We answered each query using the original signal and with wavelet reconstructions of the signal at each of the 13 possible levels. The differences are errors. In the following, we summarize the characteristics of these errors using summary statistics (mean and variance), distributional properties as measured by histograms, and autocovariance, all as a function of level and query interval length.

The main result of our evaluation is that it is possible to answer the stream and interval queries described in Section 2 on host load signals with low levels of error by using drastically compressed wavelet representations corresponding to different measurement rates. The amount of compression varies according to the granularity of the query—the coarser the granularity, the greater the compression that can be achieved with a given amount of error.

A note on presentation: Most of our results are presented using graphs similar in form to that of Figure 7(a). While these graphs show different dependent variables, they all use the same three interrelated independent variables: level,

compression, and normalized peak frequency that is resolved. The lower x-axis shows the highest level used in the analysis to reconstruct the signal. Level 13 corresponds to using all the levels (perfect reconstruction.) The upper x-axis shows compression and the normalized peak frequency (i.e., 1 corresponds to $f_s/2$) that is resolved given the level. Our purpose is to help the reader to keep in mind the exponential relationship here—eliminating one level *halves* the peak frequency we can resolve and *doubles* the compression.

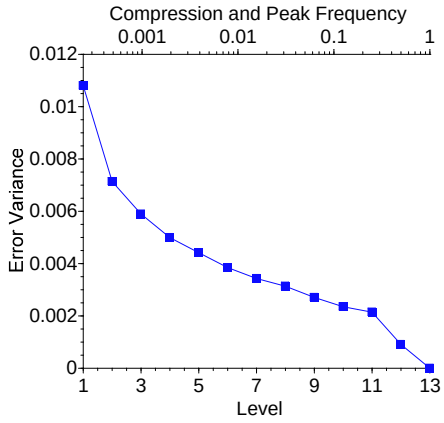
4.1. Streams

For the streaming queries, we streamed the input segments through the structure shown in Figure 2, picking different levels corresponding to a particular compression rate and peak frequency. The absolute error for this type of query is simply the error signal $\langle e_i \rangle$ described in Section 2. The mean absolute error is zero for all of the traces, even for the most active of the traces, axp0. This result tells us that the lossy compression of our wavelet-based approach does not introduce any systematic bias into stream queries.

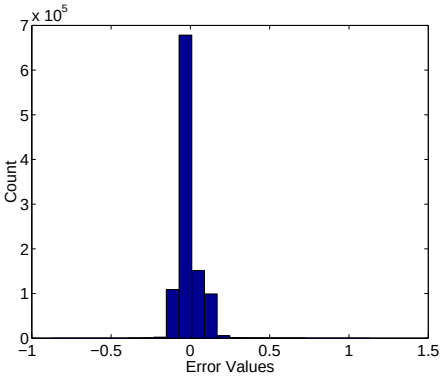
Figure 7(a) shows that the variance of the absolute error rapidly decays as we add levels. All of the traces show a decaying error variance as levels are added. The tlab-03 trace has a gradual decay—it flattens out around level 7 and 8 and then continues to decay towards zero as the number of levels is increased. We suspect that this characteristic is due to high activity in the frequency band $f_s/128$ to $f_s/64$. However, an FFT of the trace was unenlightening and resembled that of pink noise. The results for the axp0 trace decay much faster than all the other traces, but the magnitude of the error variance is higher. This is because a more active host load trace has a substantial amount of its information in the high frequency portion of its frequency spectrum, and thus more levels are needed before this information is captured well.

Figure 8 shows how the relative error variance decays as we add more levels. The relative error variance is defined as the ratio of the absolute error variance to the variance of the input signal. Even with the six lowest levels (representing fewer than 1% of the total number of coefficients), we can achieve a relative error of less than 20%. It is important to note that the shape of this figure is the same as that in Figure 7(a), but inversely scaled by the input signal variance.

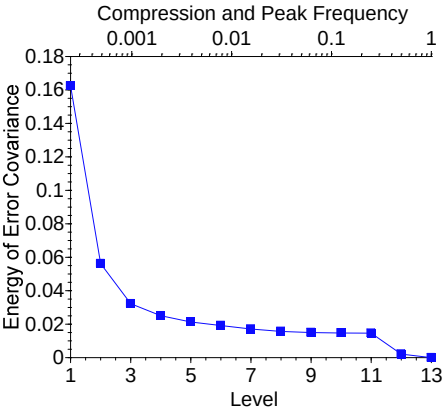
Figure 7(b) shows a typical error histogram (for level 8). Not only does the variance rapidly decay, but errors follow a near-Gaussian distribution. At level 8, we are only using 3% of the coefficients to represent the measurement stream, and yet almost all errors are less than 0.2. We see similar behavior for the other traces, except for axp0, which has a wider Gaussian, very much in line with the higher variance of that trace. The variance of the distribution quickly



(a) Absolute Error Variance



(b) Absolute Error Histogram



(c) Energy in error auto-covariance

Figure 7. Absolute error variance and energy in auto-covariance function for streaming queries as a function of reconstruction level, and representative histogram of errors.

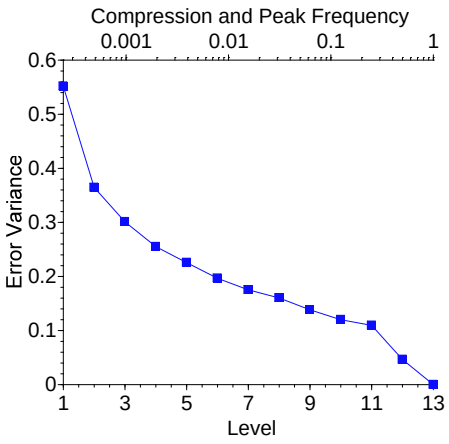


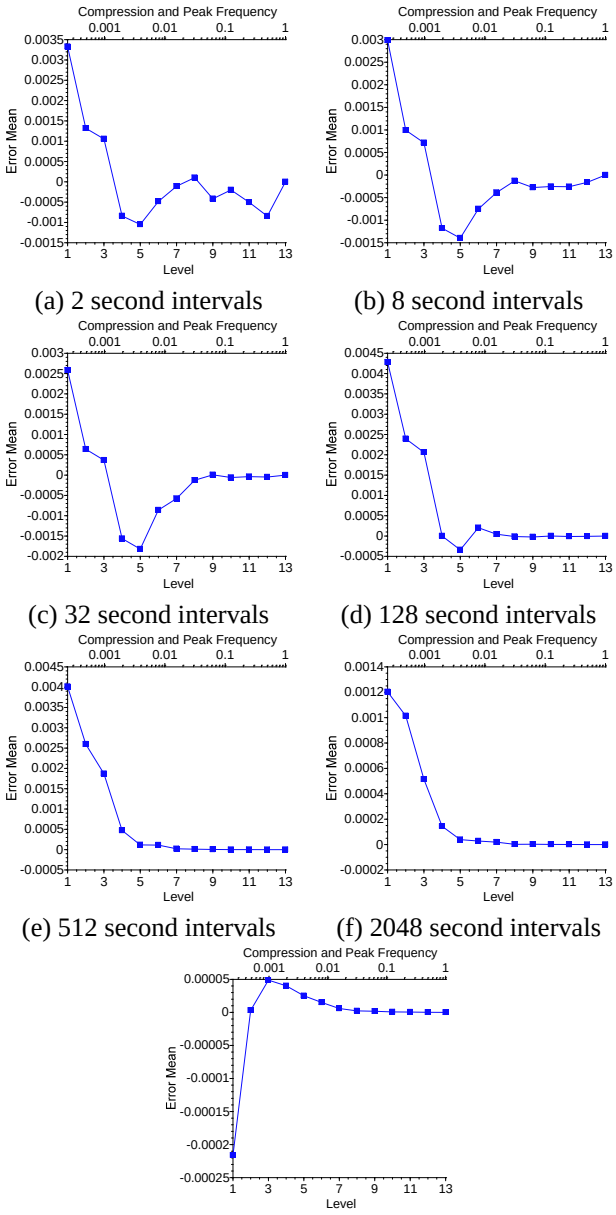
Figure 8. Relative error variance for streaming queries as a function of reconstruction level.

declines as levels are added.

The error is not strongly autocorrelated and what autocorrelation there is rapidly decays as we add levels. This can be seen from Figure 7(c), which shows the energy of the autocovariance. However, some of our more active traces do show significant autocorrelation until enough levels are added. For example, the axp0 trace requires 6 levels before the autocovariance approaches that of uncorrelated noise. Level 6 corresponds to fewer that 1% of the coefficients.

The manchester-2, sahara and themis traces present results that closely resemble those in the axp7 figures that we discussed. The axp0 trace generally requires more levels to drive our various error measures to similar low points. This is probably because axp0 has a lot more activity and high frequency information. The tlab-03 trace uses a different measure (Windows 2000 CPU usage instead of Digital Unix 5-second load) and thus its results are not directly comparable. However, the shape of the tlab-03 results is axp7's.

For all the traces that we analyzed the overall story is the same. The number of levels needed to describe the signal with low error is usually quite small, affording considerable compression. Furthermore, errors are normally distributed and i.i.d., which should make it easy to estimate the error variance for stream queries, as well as confidence intervals for interval queries.



(a) 2 second intervals (b) 8 second intervals

(c) 32 second intervals (d) 128 second intervals

(e) 512 second intervals (f) 2048 second intervals

(g) 8192 second intervals

Figure 9. Absolute error mean as a function of level for interval queries of different lengths.

4.2. Intervals

The evaluation of interval queries is considerably more complex than stream queries because the length of the interval (N) is an additional independent variable that must be accounted for. To study its effect, we chose intervals of sizes 2, 8, 32, 128, 512, 2048, and 8192 seconds.

For interval queries, we choose random intervals within our segments. We compute the average over the interval using both the original signal and the signal reconstructed using a different numbers of levels. The difference between these averages is the error that we consider here. Unlike with the stream query, the error is not a signal, but rather a sample. Therefore we don't include any autocorrelation results in this section.

Figure 9 shows the mean of the absolute error (the av-

erage error) as a function of level for the different interval lengths. The mean is close to zero, and is almost always less than a 100th of the input signal mean. For less active traces, such as manchester-2, it is even smaller. For interval queries less than a minute in duration, it takes about 7 levels for the error mean to converge to zero. This many levels corresponds to using fewer than 2% of the total number of wavelet coefficients in the reconstruction. The frequency information that is retained at this level is equivalent to $f_s/128$. For queries less than a minute long on our most active trace, axp0, the error mean converges to zero after including 9 levels (only 6% of the coefficients) in the reconstruction. Again, axp0 has more high frequency content than the other traces.

Figure 10 shows the variance of the absolute error as a function of level for the different interval lengths. The variance rapidly declines to zero as we add levels. A constraint on the variance translates to a given number of required levels. Each unnecessary level we eliminate *halves* the number of coefficients that must be communicated.

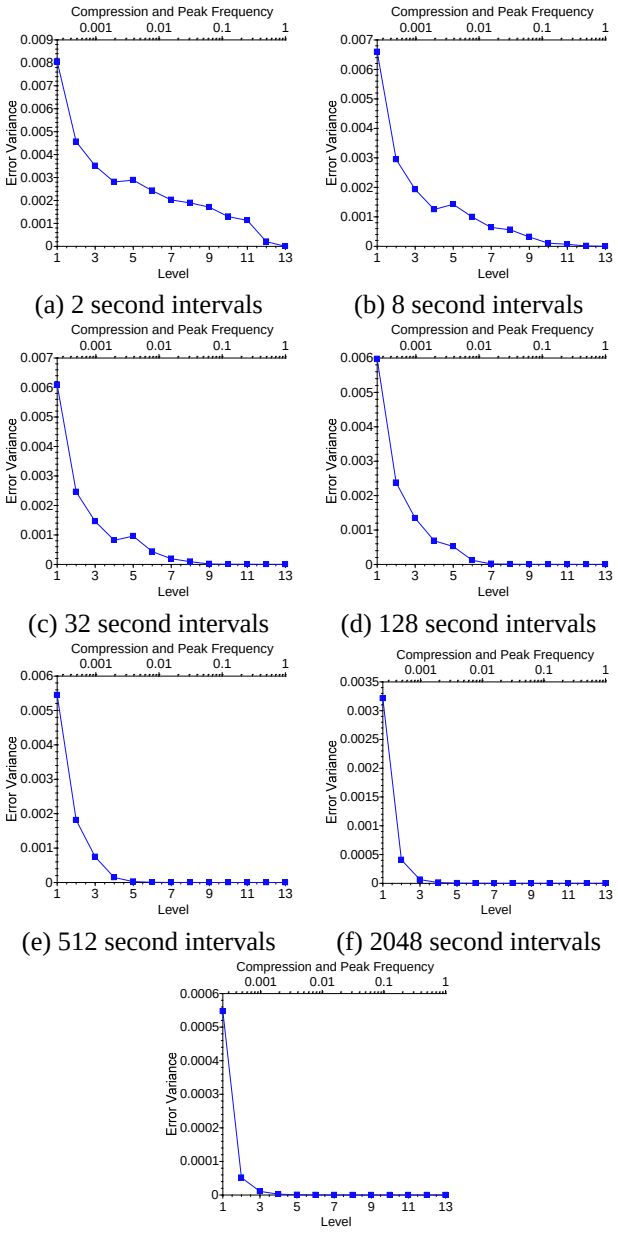
In Figure 10(f), we can see that 4 levels are sufficient to answer queries for 2048 second intervals. Using 4 levels instead of 13 leads to nearly 1000:1 compression. As the interval length of the query increases, fewer levels are required to sufficiently characterize the input signal.

For more active traces and a query interval shorter than one minute, the absolute error variance converges to zero much faster than that seen with axp7 (Figures 10(a)-(c).) On the other hand, for queries longer than a minute, the variance on inactive traces like axp7 (Figures 10(d)-(g)) converges to zero faster than those on our more active traces. For example, using query intervals longer than 2 hours (length 8192), axp0 converges to zero after only 7 levels while axp7 converges to zero after only 4 levels. Again, we blame the high frequency content in axp0. In general, the absolute error variance is 1/10th of the variance of the input signal after 7 levels.

Figure 11 shows histograms of the error (at reconstruction using level 5 and below) for the different interval lengths. It is important to note that the scales along the x-axis and y-axis are different for each of the plots and the scale for plot (g) is drastically lower than the rest. Notice that the error is usually quite Gaussian once a sufficient level or interval length has been achieved.

There is a tradeoff between the interval length requested and the level that is required to satisfy the query with small error. If the interval request is long (greater than a minute), the query can be satisfied using a lower level. Likewise, if the interval request is short (less than a minute), a higher level should be used to maintain low error variability. In our more active trace, axp0, the histogram at level 5 is not Gaussian until the interval query is 512 seconds long.

The results for the sahara and themis traces are similar to those shown for axp7. The results of the trace manchester-2 at query interval lengths less than a minute in duration, are substantially better than the other traces because of its inactivity. axp0 requires longer duration interval queries, or more levels in the reconstruction to maintain low error metrics. This is again because of its high activity, high usage characteristics. The results of the trace tlab-03 are on par



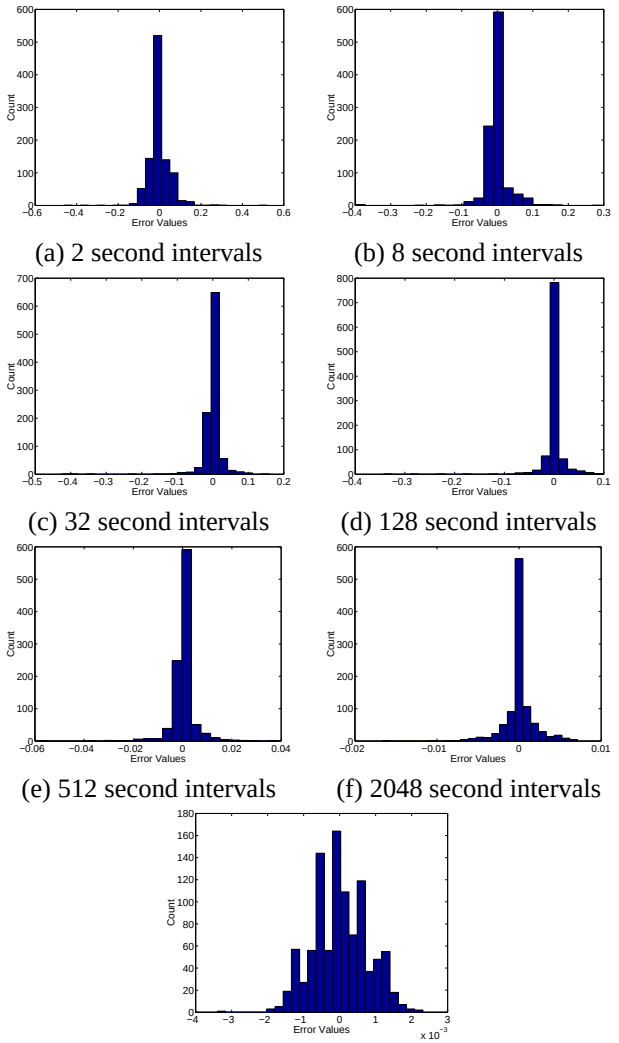
(a) 2 second intervals (b) 8 second intervals

(c) 32 second intervals (d) 128 second intervals

(e) 512 second intervals (f) 2048 second intervals

(g) 8192 second intervals

Figure 10. Absolute error variance as a function of level for interval queries of different lengths



(a) 2 second intervals (b) 8 second intervals

(c) 32 second intervals (d) 128 second intervals

(e) 512 second intervals (f) 2048 second intervals

(g) 8192 second intervals

Figure 11. Histograms of error variance at level five for interval queries of different lengths

with those of axp7.

A note on relative error: We had originally intended to include a discussion of relative error mean and variance for interval queries. The relative error mean is defined as the ratio of the absolute error mean to the mean of the input signal over the interval. The problem with calculating this metric is that there are intervals in which the mean of the input signal is zero (no activity in the trace). The relative error variance is defined as the ratio of the absolute error variance to the variance of the input signal over the interval, introducing similar divide-by-zero problems when an interval has no variance. We did compute these metrics using only intervals with non-zero mean and variance, producing numbers in-line with results presented here. However, selecting non-zero samples amounts to a biased sampling process and hence we are uncomfortable presenting the graphs here.

5. Conclusions and future work

We introduced a query model for resource signals that explicitly incorporates time-scale as a parameter to both stream and interval queries. We demonstrated how to build the model on top of a wavelet-domain representation of resource signals. This computationally inexpensive approach communicates just enough data to answer the query, often leading to substantial levels of compression with little effect on accuracy. We reached this conclusion by evaluating our approach on host load signals.

The wavelet representation not only helps us to decouple sensors and applications, but it is also an enabler of other powerful techniques for processing resource signals. For example, we believe that further compression gains with no loss of accuracy can be achieved by using wavelet denoising, a well-known technique in the signal processing community that we discussed at the end of Section 3. We have experimented with de-noising host load signals and have found that the wavelet coefficients can be further compressed by a factor of 10 without a substantial increase in error. These results will be forthcoming.

The wavelet representation also provides a natural way to gradually increase compression as the signal ages, simply by throwing away more levels or increasing the threshold in de-noising. We are currently studying this scheme.

Improved compression allows for queries over longer intervals of time using the same amount of storage space or network bandwidth. Application schedulers will be able to better validate a machine by looking at its performance over months of time instead of days.

We suspect that wavelet-domain representations will lead to improved prediction of resource signals due to the uncorrelated nature of the wavelet transformation. By predicting each level separately, we may be able to do much better, especially for long-range predictions, than simply considering the time-domain signal. We are currently implementing the wavelet representation and predictors in the RPS system [5].

At this point, we have only applied our approach to Digital Unix host load signals and to CPU availability signals on Windows 2000. We plan to apply the approach to signals that measure network properties, perhaps using Remos [9] as the producer of the signals.

The application of wavelet-based techniques to resource signals appears to have tremendous potential.

References

[1] F. Berman and R. Wolski. Scheduling from the perspective of the application. In *Proceedings of the Fifth IEEE Symposium on High Performance Distributed Computing HPDC96*, pages 100–111, August 1996.

[2] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics (SIAM), 1999.

[3] P. A. Dinda. The statistical properties of host load. *Scientific Programming*, 7(3,4), 1999. A version of this paper is also available as CMU Technical Report CMU-CS-TR-98-175. A much earlier version appears in LCR '98 and as CMU-CS-TR-98-143.

[4] P. A. Dinda. *Resource Signal Prediction and Its Application to Real-time Scheduling Advisors*. PhD thesis, School of Computer Science, Carnegie Mellon University, May 2000. Available as Carnegie Mellon University Computer Science Department Technical Report CMU-CS-00-131.

[5] P. A. Dinda and D. R. O'Hallaron. An extensible toolkit for resource prediction in distributed systems. Technical Report CMU-CS-99-138, School of Computer Science, Carnegie Mellon University, July 1999.

[6] P. A. Dinda and D. R. O'Hallaron. Host load prediction using linear models. *Cluster Computing*, 3(4), 2000. An earlier version of this paper appeared in HPDC '99.

[7] A. Feldman, A. C. Gilbert, and W. Willinger. Data networks as cascades: Investigating the multifractal nature of internet WAN traffic. In *Proceedings of ACM SIGCOMM '98*, pages 25–38, 1998.

[8] M. W. Knop, P. K. Paritosh, P. A. Dinda, and J. M. Schopf. Windows performance monitoring and data reduction using watchtower and argus. Technical Report NWU-CS-01-6, Department of Computer Science, Northwestern University, June 2001.

[9] B. Lowekamp, N. Miller, D. Sutherland, T. Gross, P. Steenkiste, and J. Subhlok. A resource monitoring system for network-aware applications. In *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 189–196. IEEE, July 1998.

[10] P. Moulin and J. Liu. Analysis of multiresolution image denoising schemes usin g generalized-gaussian priors. In *Proc. IEEE TFTS Symposium*, pages 633–636, Pittsburgh, PA, Oct. 6-9 1998.

[11] D. E. Newland. *An Introduction to Random Vibrations, Spectral and Wavelet Analysis*. Addison Wesley Longman Limited, 1993.

[12] A. V. Oppenheim, A. S. Willsky, and I. T. Young. *Signals and Systems*. Prentice Hall, 1983.

[13] V. J. Ribeiro, R. H. Riedi, M. S. Crouse, and R. G. Baraniuk. Simulation of non-gaussian long-range-dependent traffic using wavelets. In *Proceedings of ACM SIGMETRICS '99*, pages 1–12, May 1999.

[14] J. M. Schopf and F. Berman. Stochastic scheduling. In *Proceedings of SuperComputing '99*, 1999. Also available as Northwestern University, Computer Science Department Technical Report CS-99-3, or http://www.cs.nwu.edu/∼jms/Pubs/TechReports/sched.ps.

[15] R. Wolski. Forecasting network performance to support dynamic scheduling using the network weather service. In *Proceedings of the 6th High-Performance Distributed Computing Conference (HPDC97)*, pages 316–325, August 1997. extended version available as UCSD Technical Report TR-CS96-494.

[16] G. Wornell. *Signal Processing with Fractals: A Wavelet-Based Approach*. Prentice Hall, 1995.