Timescale Stream Statistics for Hierarchical Management

Chen Ding and Pengcheng Li
University of Rochester

> The voyage of the best ship is a zigzag line of a hundred tacks. See the line from a sufficient distance, and it straightens itself to the average tendency.
>
> R. W. Emerson *Self-reliance*

This short paper introduces a type of measurement called timescale statistics and shows its use in optimizing resource allocation especially resources organized in a hierarchy. Through two examples, it extracts a common framework called Optimal Hierarchy Sharing (OHS).

Timescale Stream Statistics The definition of a stream is given in the final report of the STREAM 2015 workshop, which states up front that "We define a stream as a possibly unbounded sequence of events. Successive events may or may not be correlated and each event may optionally include a timestamp." Adding to this definition, we can define a time window by the start and the end time that includes the events between these two times. The length of a window is its end time minus its start time plus one. In the absence of timestamps, or in addition to them, a logical time can be assigned to each event, e.g. the index of the event in the stream. A time window is also called a time interval.

A timescale metric, $f(x)$, is a function parameterized by the timescale. A timescale x is a length of time, and $f(x)$ is the average behavior of all time windows of length x. The function $f(x)$ shows the average behavior at all timescales, i.e. for all $x \geq 0$.

An example is temperature variation. Given the average air temperature each day for several years as a stream of numbers, we can quantify the temperature fluctuation at different timescales. For each period of x consecutive days, the peak variation is the highest daily temperature minus the lowest. When $x = 1$, we have $pv(1) = 0$. When $x = 2$, $pv(2)$ is average temperature difference between all two-consecutive days. Similarly for any $x > 2$, $pv(x)$ is the average of the peak variations in all periods of x consecutive days. The metric $pv(x)$ shows complete timescale dynamics, i.e. the expected greatest change in daily temperature, in any number of days, for example, a week time or a month time. A timescale metric examines all time periods and therefore avoids the data bias if it were to measure just some periods, e.g. calendar weeks or months.

Existing metrics are based on statistics on a single timescale, e.g. instant (last moment or last window) or cumulative (from the start till now). Timescale metrics add an important new dimension, which is useful in resource allocation, as shown next by two examples.

Timescale Locality for Hierarchical Cache Memory Cache hierarchy is a hierarchy of fast memories. The cache memory at each level is automatically managed and shared among parallel tasks. The performance is measured by the miss ratio, i.e. the average portion of memory accesses that are misses.

Footprint $fp(x)$ is a timescale locality metric that measures the average working-set size of all windows of length x. Footprint is used to compute the miss ratio of all cache sizes, i.e. the miss ratio curve. From miss ratio curves, we can optimize cache sharing by techniques such as optimal program symbiosis [4] and optimal cache partition [1] for CPU cache and optimal allocation in the in-memory key-value store Memcached [3]. Footprint can be measured in linear time. The measurement costed less than 0.1 second per program [4] and could be performed entirely online [3].

Timescale Memory Demand for Hierarchical Heap Memory Concurrent memory allocation often uses a hierarchy of heaps. The free memory at each level is automatically managed and shared among parallel threads. The main cost is an inter-heap memory fetch, i.e. fetching free memory from another heap. The key metric is the fetch ratio,

i.e. the average portion of memory allocations that fetch memory from another heap. The fetch cost increases with concurrency because of the atomicity control of the meta-data.

A recent invention (not published) is the timescale metric $pd(x)$, which is the peak demand over timescale x. The peak demand is used to compute the fetch ratio $fr(r)$ for reserve (free memory) of size r for all $r \geq 0$. The fetch ratio curve enables optimal partitioning of the heap memory, just as miss ratio curve enables it for cache memory.

A Conjecture Resource is often organized hierarchically. In a parallel system, the resource hierarchy is shared. Cache and heap memories are two examples of a common framework for optimal hierarchical sharing (OHS). Others may include computer networks, power supplies and cooling. From the two examples, we make the conjecture that

Timescale stream statistics enables optimal sharing of hierarchical resources

In other words, OHS can be solved by timescale statistics. First, a timescale metric captures the demand of each parallel task at all timescales. Second, the timescale demand is used to predict their performance at all resource allocations. Third, the resource is optimally allocated among parallel tasks. In our first example, the timescale footprint predicts the performance of all cache sizes, which we use to optimally partition the cache at all levels. In the second example, the timescale memory demand predicts the heap performance at all free memory sizes, which we use to optimize memory allocation at all heap levels.

OHS is enabled by timescale statistics. It connects program behavior, which is a property of time, to performance, which is a property of resource allocation.

A property of OHS is monotonicity: the (optimized) performance of a parallel program does not decrease when it is given more resource. It does not hurt when using more resource. For cache memory, the violation of monotonicity is known as the Belady anomaly. Such anomaly is possible with unmanaged cache sharing but not when the shared cache is optimally allocated.

> The true poem is the poet's mind; the true ship is the ship-builder.
>
> R. W. Emerson *History*

There are a number of interesting questions to study to examine the conjecture:

New OHS Applications Locality optimization is generally useful. One possible application is the static or run-time optimization of stream programs, i.e. as a cost model for use in the optimization of stream programs [2]. More convincing is whether the framework is useful beyond the two OHS examples. A potential problem is that a timescale measure at x is the average of all length-x windows. It gives the expected behavior of a randomly chosen window, but not the worst-case behavior. In the two examples, the goal was maximizing utilization or throughput, hence the average performance of all windows. It is an open question whether timescale metrics should measure more than just the average behavior.

Timescale Algorithms As new timescale metrics are defined, efficient algorithms are needed to compute the all-window statistics, whether it is for the average or other types of statistics. The total number of windows is quadratic to the length of a stream. Linear time algorithms have been found for the two OHS examples and successfully used for online optimization. These algorithmic solutions are not trivial, and we should expect to face significant difficulties in new problems. Another need of the algorithm development is to provide formal proofs of monotonicity. For example, based on the linear-time solution, the footprint can be shown concave, and hence the miss ratio monotone [5].

Visualization and Human in the Loop As a function, a timescale measure provides a way to visualize the timescale behavior of an application, task, or task group. It is an interesting question whether the average behavior can help a user better understand a program and enable more symbiotic interaction between the user and the machine.

References

[1] J. Brock, C. Ye, C. Ding, Y. Li, X. Wang, and Y. Luo. Optimal cache partition-sharing. pages 749–758, 2015.

[2] M. Hirzel, R. Soulé, S. Schneider, B. Gedik, and R. Grimm. A catalog of stream processing optimizations. *ACM Computing Surveys*, 46(4):46:1–46:34, 2013.

[3] X. Hu, X. Wang, Y. Li, L. Zhou, Y. Luo, C. Ding, S. Jiang, and Z. Wang. LAMA: Optimized locality-aware memory allocation for key-value cache. In *Proceedings of USENIX ATC*, 2015.

[4] X. Wang, Y. Li, Y. Luo, X. Hu, J. Brock, C. Ding, and Z. Wang. Optimal footprint symbiosis in shared cache. In *Proceedings of CCGrid*, pages 412–422, 2015.

[5] X. Xiang, C. Ding, H. Luo, and B. Bao. HOTL: a higher order theory of locality. In *Proceedings of ASPLOS*, pages 343–356, 2013.