Maxeler Apps
Correlation

MAXELER
Technologies
MAXIMUM PERFORMANCE COMPUTING

Dec 2014

Correlation

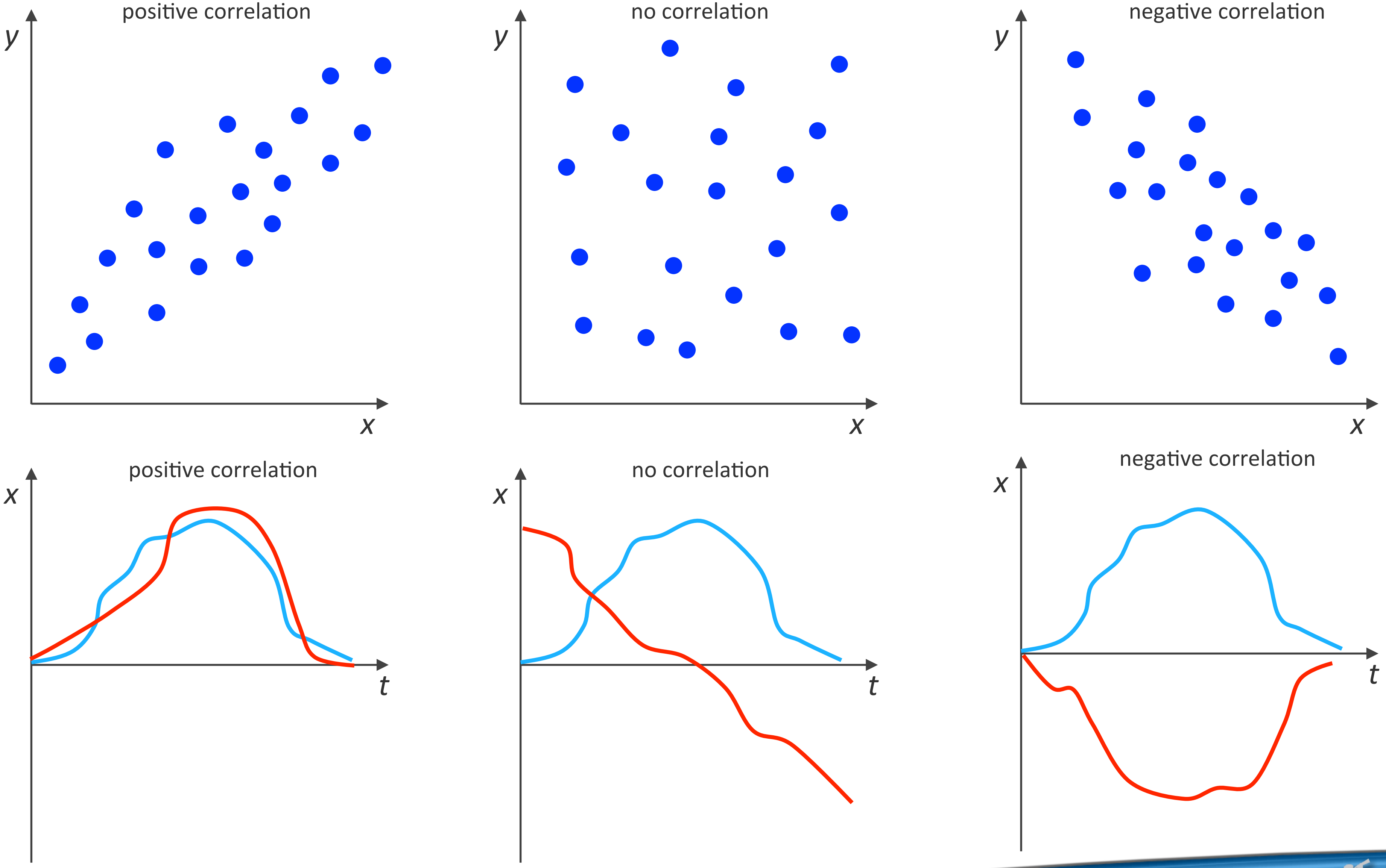
- Statistical method to analyze the relationship between two random variables.
- Intuitive interpretation: “How similar do two variables behave? Do curves have similar shapes? ”
- Various methods exist to measure relationship between variables.
- Pearson correlation measures *linear relationship*. Produces values between 1 and -1:

  1: exactly identical behavior

  0: no relationship

  -1: exactly opposite behavior
- Correlation not imply direct causality!

Examples



positive correlation
y
x
no correlation
y
x
negative correlation
y
x
positive correlation
x
t
no correlation
x
t
negative correlation
x
t

Pearson product-moment correlation

- Product-moment correlation: Calculate mean (i.e. first moment) of product of mean-adjusted values:

$$\rho_{X,Y} = \frac{E\left[\left(X-\mu_X\right)\left(Y-\mu_Y\right)\right]}{\sigma_X \sigma_Y}$$

with

E	expected value
μ	mean value
σ	standard deviation

- For sample data, calculate as:

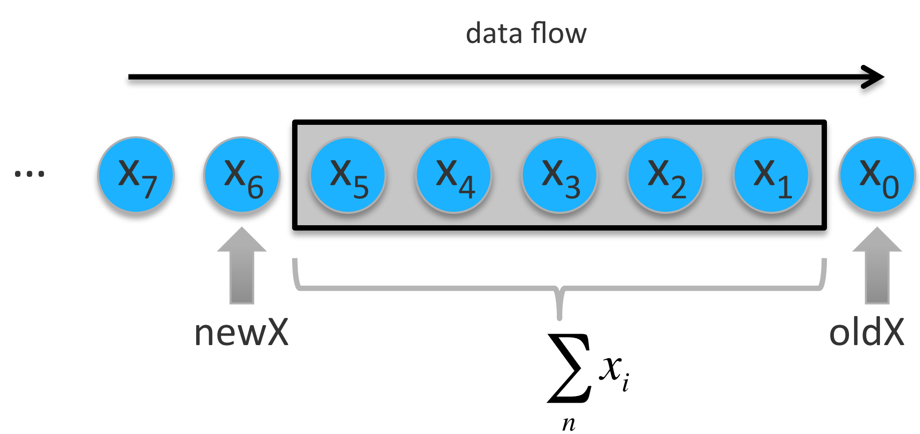
$$r_{x,y} = \frac{\sum_{i=1}^{n}\left(x_i-\overline{x}\right)\left(y_i-\overline{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2}\sqrt{\sum_{i=1}^{n}\left(y_i-\overline{y}\right)^2}} = \frac{n\sum_{i=1}^{n}x_i y_i - \sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i}{\sqrt{n\sum_{i=1}^{n}x_i^2-\left(\sum_{i=1}^{n}x_i\right)^2}\sqrt{n\sum_{i=1}^{n}y_i^2-\left(\sum_{i=1}^{n}y_i\right)^2}}$$

How to compute a correlation in practice

- Computation of correlation is based on 5 sums:

$$\text{sumXY: } \sum_n x_i y_i \quad \text{sumX: } \sum_n x_i \quad \text{sumY: } \sum_n y_i \quad \text{sumX2: } \left(\sum_n x_i\right)^2 \quad \text{sumY2: } \left(\sum_n y_i\right)^2$$

- Typical application involves continuous correlation of time series: use sliding window approach.



No need to recompute entire sum, add next element and subtract previous:

```
sumX += newX - oldX;
```

For every time step, complexity of computing sums = O(1) regardless of window size!

Correlation between many time series

- Application in finance requires correlations between 200 to 6000 time series.
  - Complexity arises from number of time series *m* to be correlated, not from number of data elements *n* to be summed in sliding window.

O(m[2]) sums_xy

O(m) sums

O(m[2]) correlations

$$r_{x,y} = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}}$$

O(m) sums_sq

MAXELER Technologies

C implementation

```
for (uint64_t s=0; s<numTimesteps; s++) {          // loop over all data
    index_correlation = 0;
    for (uint64_t i=0; i<numTimeseries; i++) {        // precompute sums and sums of squares
        double old = (s>=windowSize ? data[i][s-windowSize] : 0);
        double new = data[i][s];
        sums[i] += new - old;              // compute sum in current window
        sums_sq[i] += new*new - old*old;       // compute sum of squares in current window
    }
    for (uint64_t i=0; i<numTimeseries; i++) {         // correlation outer loop
        double old_x = (s>=windowSize ? data[i][s-windowSize] : 0);
        double new_x = data [i][s];
        for (uint64_t j=i+1; j<numTimeseries; j++) {         // correlation inner loop
            double old_y = (s>=windowSize ? data[j][s-windowSize] : 0);
            double new_y = data[j][s];
            sums_xy[index_correlation] += new_x*new_y - old_x*old_y;       // sum of x*y
            correlations_step[index_correlation] = (windowSize*sums_xy[index_correlation] - sums[i]*sums[j]) /
(sqrt(windowSize*sums_sq[i] - sums[i]*sums[i])* sqrt(windowSize*sums_sq[j] -sums[j]*sums[j]));       // correlation
            indices_step[2*index_correlation] = j;
            indices_step[2*index_correlation+1] = i;
            index_correlation++;
        }
    }
}
```

Opportunities for acceleration

```
for (uint64_t s=0; s<numTimesteps; s++) {
    index_correlation = 0;
    for (uint64_t i=0; i<numTimeseries; i++) {
        double old = (s>=windowSize ? data[i][s-windowSize] : 0);
        double new = data[i][s];
        sums[i] += new - old;
        sums_sq[i] += new*new - old*old;
    }
    for (uint64_t i=0; i<numTimeseries; i++) {
        double old_x = (s>=windowSize ? data[i][s-windowSize] : 0);
        double new_x = data [i][s];
        for (uint64_t j=i+1; j<numTimeseries; j++) {
            double old_y = (s>=windowSize ? data[j][s-windowSize] : 0);
            double new_y = data[j][s];
            sums_xy[index_correlation] += new_x*new_y - old_x*old_y;
            correlations_step[index_correlation] = (windowSize*sums_xy[index_correlation] - sums[i]*sums[j]) /
(sqrt(windowSize*sums_sq[i] - sums[i]*sums[i])* sqrt(windowSize*sums_sq[j] -sums[j]*sums[j]));
            indices_step[2*index_correlation] = j;
            indices_step[2*index_correlation+1] = i;
            index_correlation++;
        }
    }
}
```

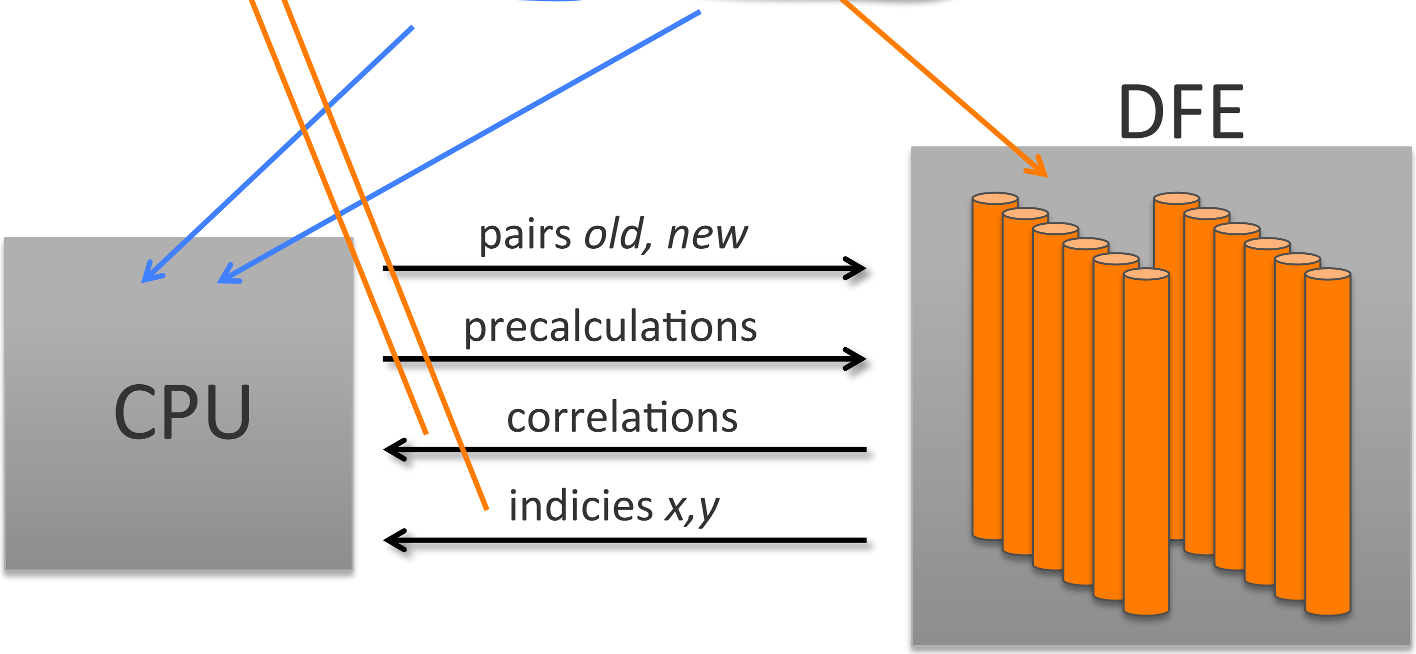
store pairs of *new* and *old* in LMEM

accelerate with DFE

precompute sums and inverse of sqrt on CPU

Split correlation on CPU and DFE

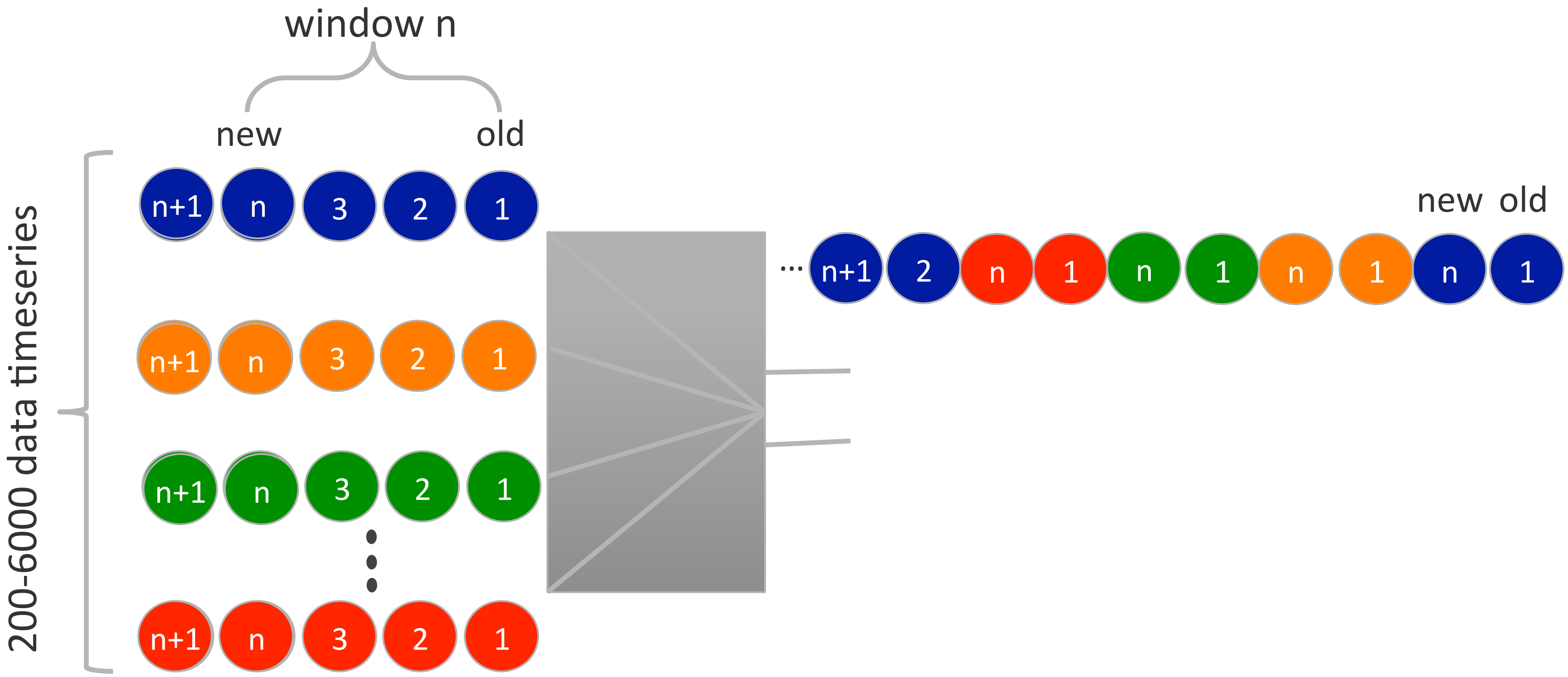
$$r_{x,y} = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2}}$$



12 pipes fill the chip

each pipe has a feedback loop of 12 ticks => 144 calculations running at the same time
see MaxCompiler tutorial on Loops and Pipelining

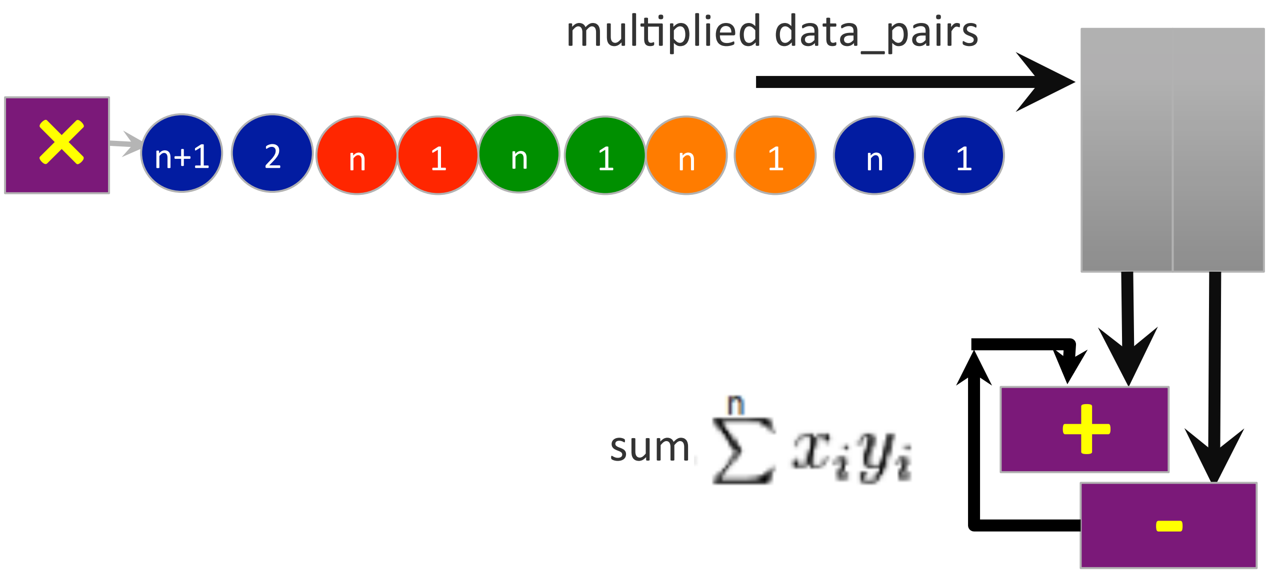
Reorder input data for DFE



- First and last element (1, n) of each window creates a data_pair (old, new), then go round robin through time series.
- Next move the window to (2, n+1), they become the new data_pair.

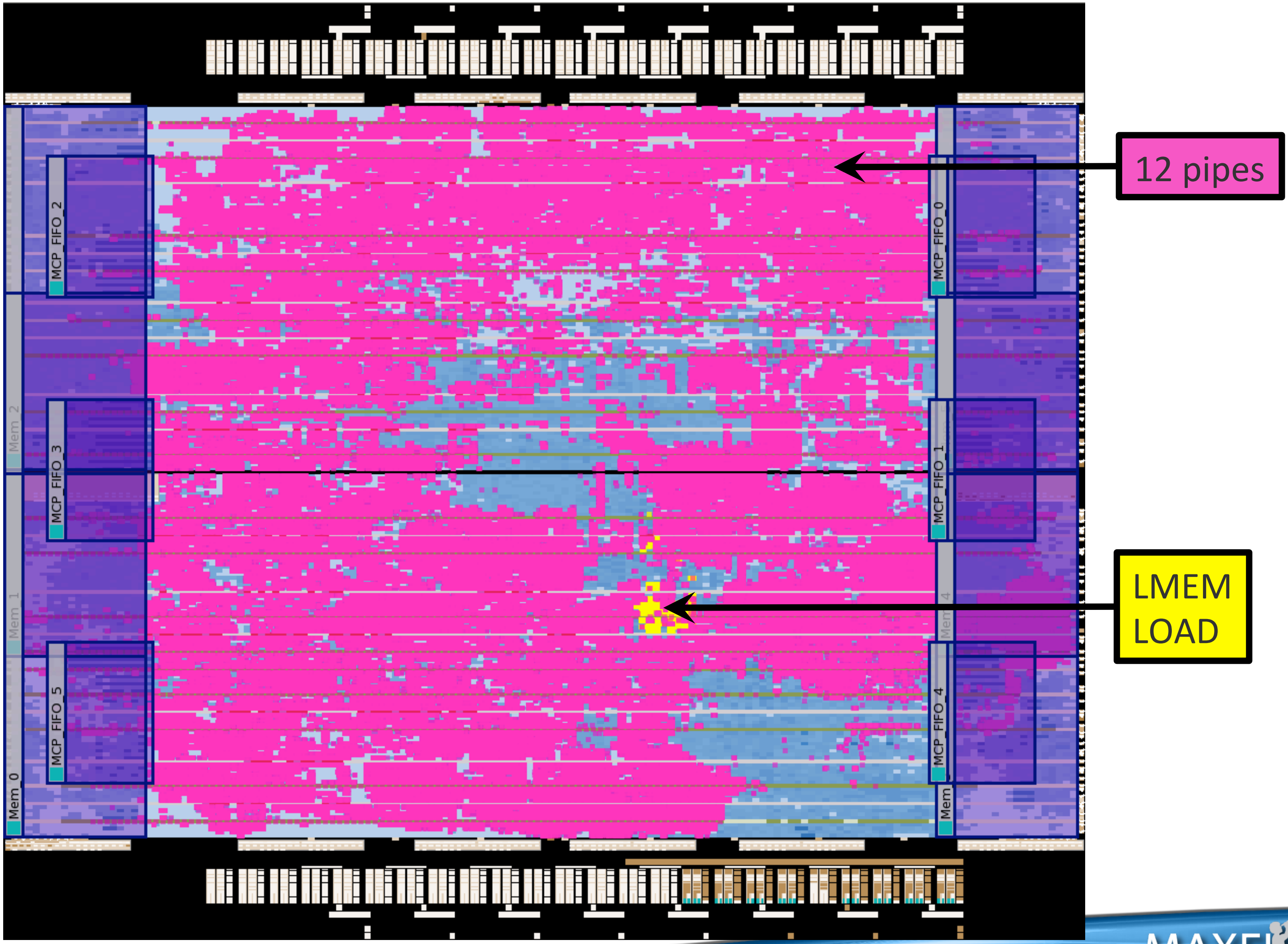
MAXELER Technologies

Accumulate interleaved data

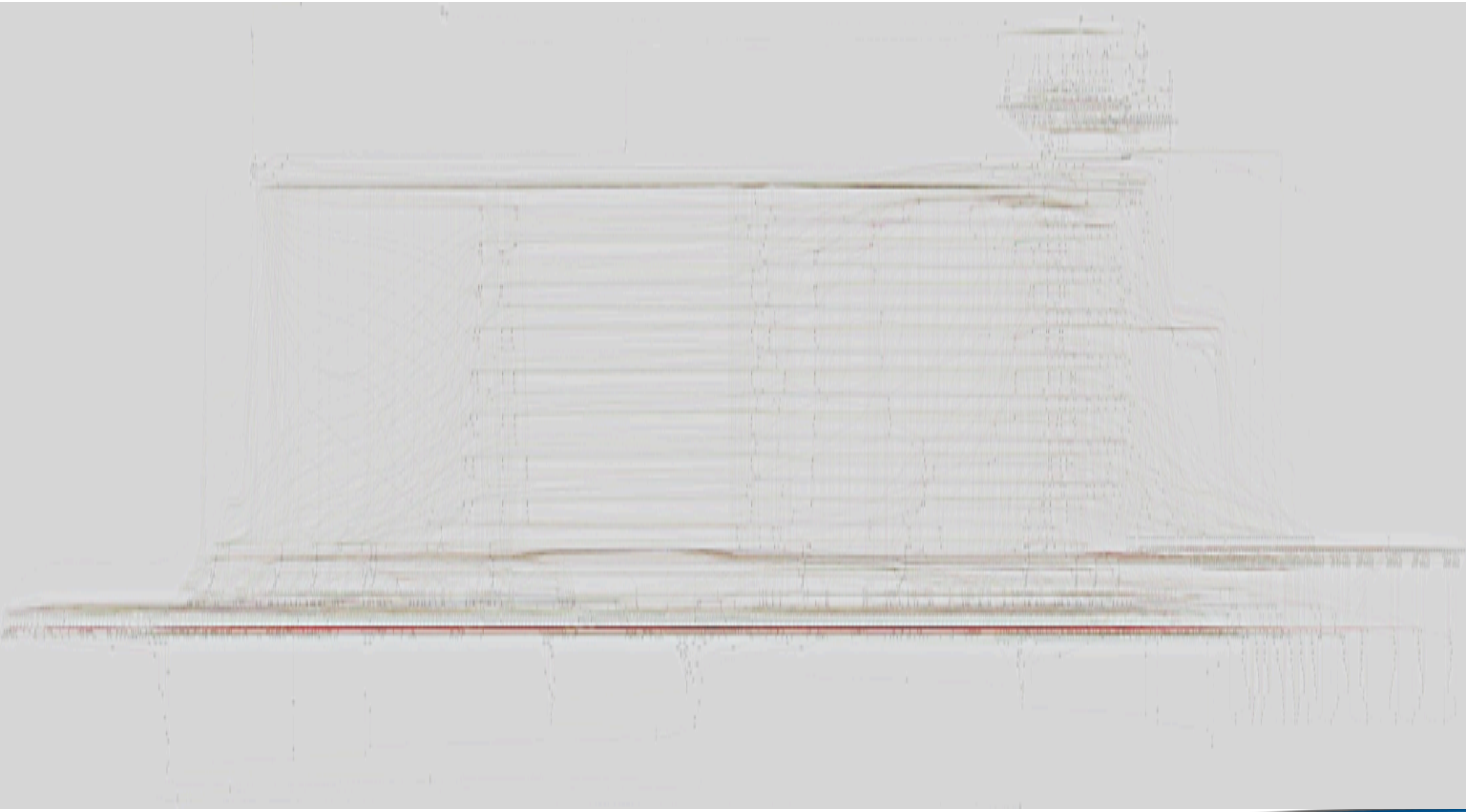


- Compute window sum by subtracting last element and adding new.
- Reordered and interleaved data maximized computation in pipes with feedback loops.

DFE engine for correlation



Data flow graph for correlation



Code example

File: correlationCPUCode.c
Purpose: calling correlationSAPI.h for correlation.max

Correlation formula:

scalar r(x,y) = (n*SUM(x,y) - SUM(x)*SUM(y))*SQRT_INVERSE(x)*SQRT_INVERSE(y)
where:
x,y, ... - Time series data to be correlated
n - window for correlation (minimum size of 2)
SUM(x) - sum of all elements inside a window
SQRT_INVERSE(x) - 1/sqrt(n*SUM(x^2)- (SUM(x)^2))

Action 'loadLMem':
[in] memLoad - initializeLMem, used as temporary storage

Action 'default':

[in] precalculations: {SUM(x), SQRT_INVERSE(x)} for all timeseries for every timestep

[in] data_pair: {..., x[i], x[i-n], y[i], y[i-n], ... , x[i+1], x[i-n+1], y[i+1], y[i-n+1], ...} for all timeseries for every timestep

[out] correlation r: numPipes * CorrelationKernel_loopLength * topScores correlations for every timestep

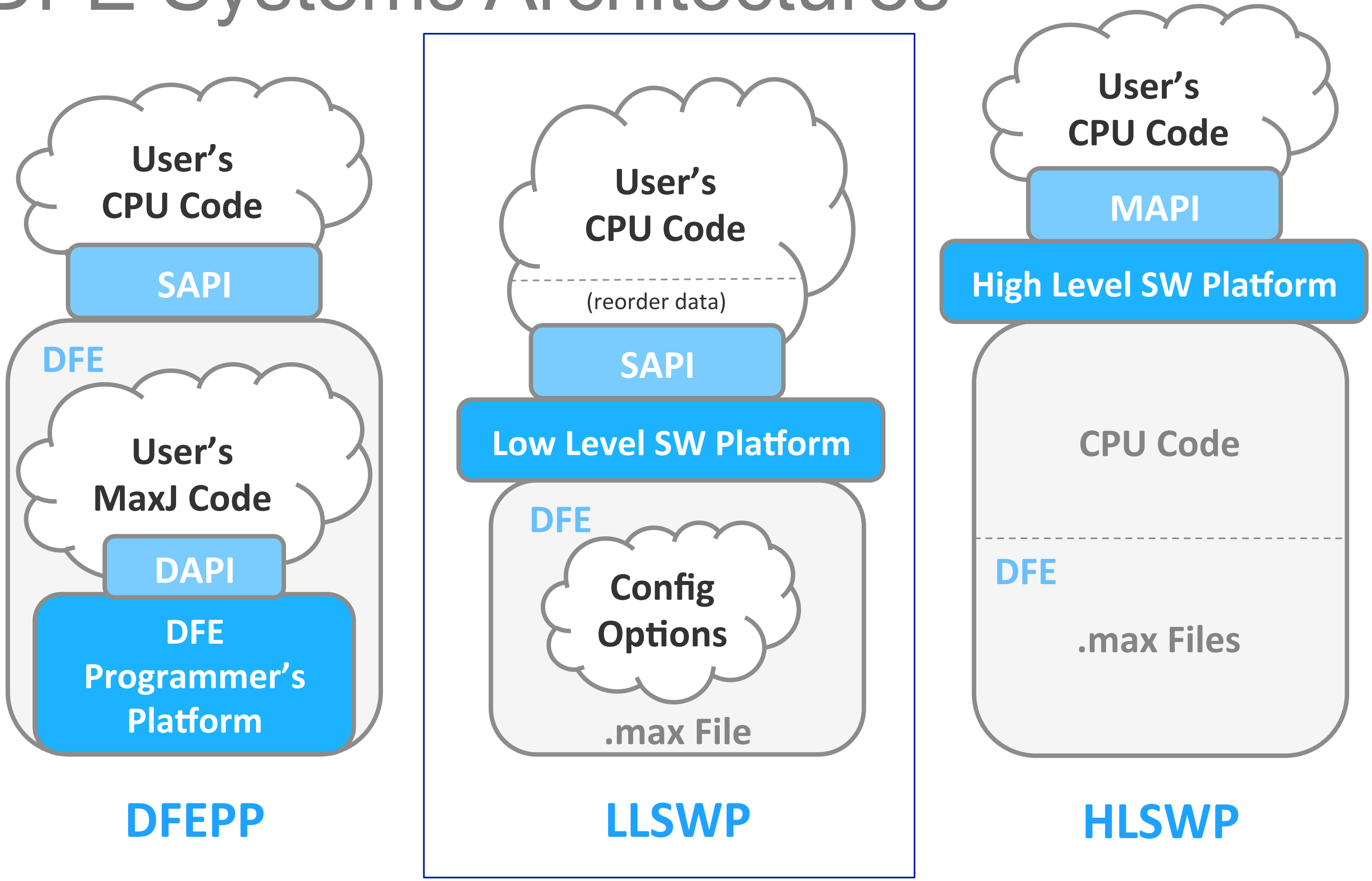
[out] indices x,y: pair of timeseries indices for each result r in the correlation stream

MAXELER Technologies

Code example – data reordering

```
// 2 DFE input streams: precalculations and data pairs
for (uint64_t i=0; i<numTimesteps; i++) {
        old_index = i - (uint64_t)windowSize;

        for (uint64_t j=0; j<numTimeseries; j++) {
                if (old_index<0) old = 0; else old = data [j][old_index];
                new = data [j][i];

                if (i==0) {
                        sums [i][j] = new;
                        sums_sq [i][j] = new*new;
                }else {
                        sums [i][j] = sums [i-1][j] + new - old;
                        sums_sq [i][j] = sums_sq [i-1][j] + new*new - old*old;
                }
                inv [i][j] = 1/sqrt((uint64_t)windowSize*sums_sq[i][j] - sums[i][j]*sums[i][j]);

                // precalculations REORDERED in DFE ORDER
                precalculations [2*i*numTimeseries + 2*j] = sums[i][j];
                precalculations [2*i*numTimeseries + 2*j + 1] = inv [i][j];

                // data_pairs REORDERED in DFE ORDER
                data_pairs[2*i*numTimeseries + 2*j] = new;
                data_pairs[2*i*numTimeseries + 2*j + 1] = old;
        }
}
```

DFE Systems Architectures



EXAMPLES

Max MPT
Video transcoding/processing

Correlation App

Risk Analytics Library
Video Encoding

Summary

- Measure relationship between time series
- Quadratic complexity with regard to number of time series
- Split computation between O(n) pre-computations (target CPU) and O(n^2) calculations (target DFE)
- Reorder and interleave data to maximize computation in DFE
- Implement parallel pipes to utilize available DFE resources