MODELING THE EFFECTS OF DATA LOCALITY

András Leitereg

ELTE Faculty of Informatics

July 11, 2019

EFOP-3.6.3-VEKOP-16-2017-00001

OUTLINE

- Motivation
- Previous work
  - The LambdaGen compiler
  - Expression transformations
- Predicting performance
  - Tensor contractions
  - (Counter) examples
  - Neural network structure
  - Results
- Conclusion

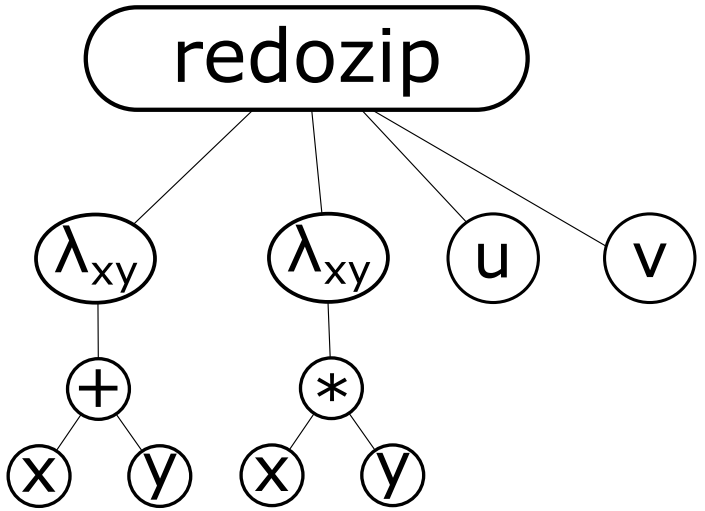
MOTIVATION

- Linear algebra
  - The core of many HPC applications
  - Clear and regular structure
- Hierarchical computations
- Hierarchical hardware:
  - Devices
  - Threads
  - Memory
- Functional vector operations
  - map, zip, reduce

THE LAMBDAGEN COMPILER

- Expression tree
  - Scalar value
  - Scalar operations
  - Data view
  - Lambda abstraction
  - Lambda application,
  - Variable
  - Zip, Redozip

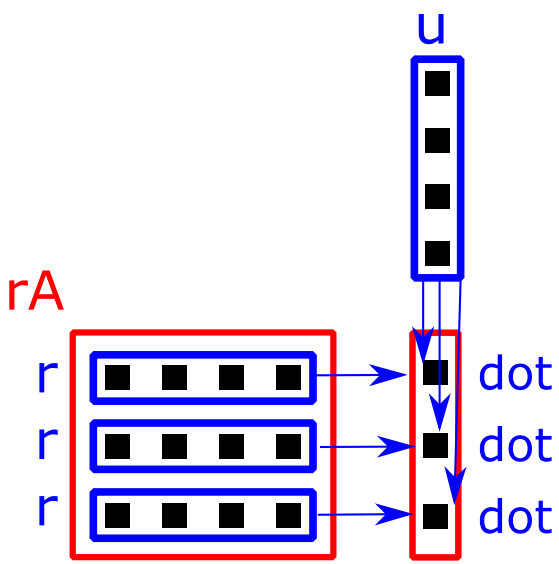
$$\sum_i u_i v_i$$

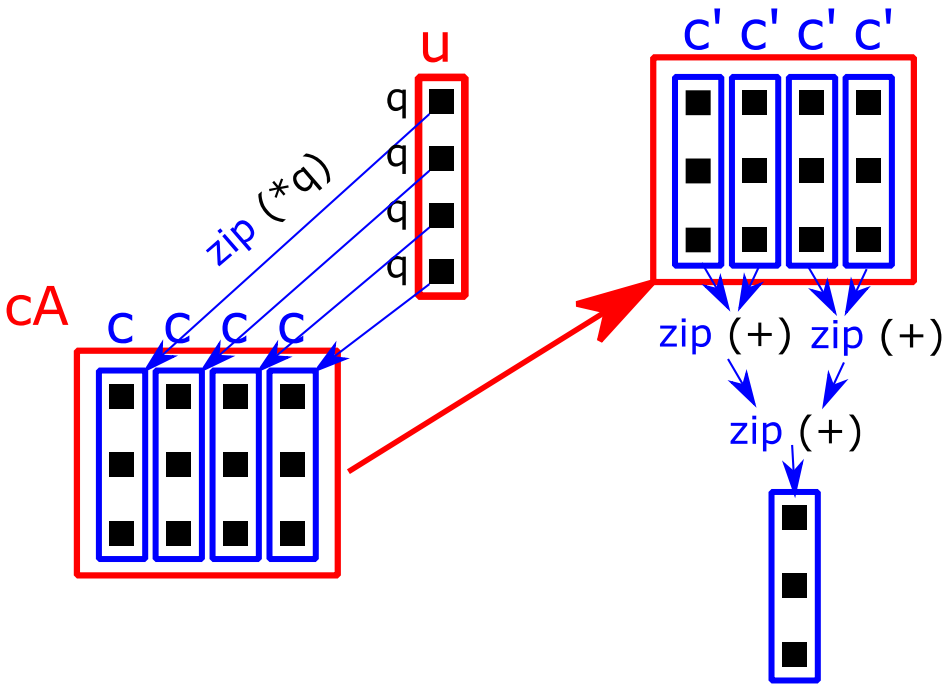


THE LAMBDAGEN COMPILER

- Passes based on recursion schemes
  - Typecheck
  - Closure conversion + lambda lifting
  - Storage allocation
  - CPU & GPU codegen
- Analyses produce annotations
- Pattern based search-replace
- See GPU Day 2017 and 2018 for more…

EXPRESSION TRANSFORMATIONS



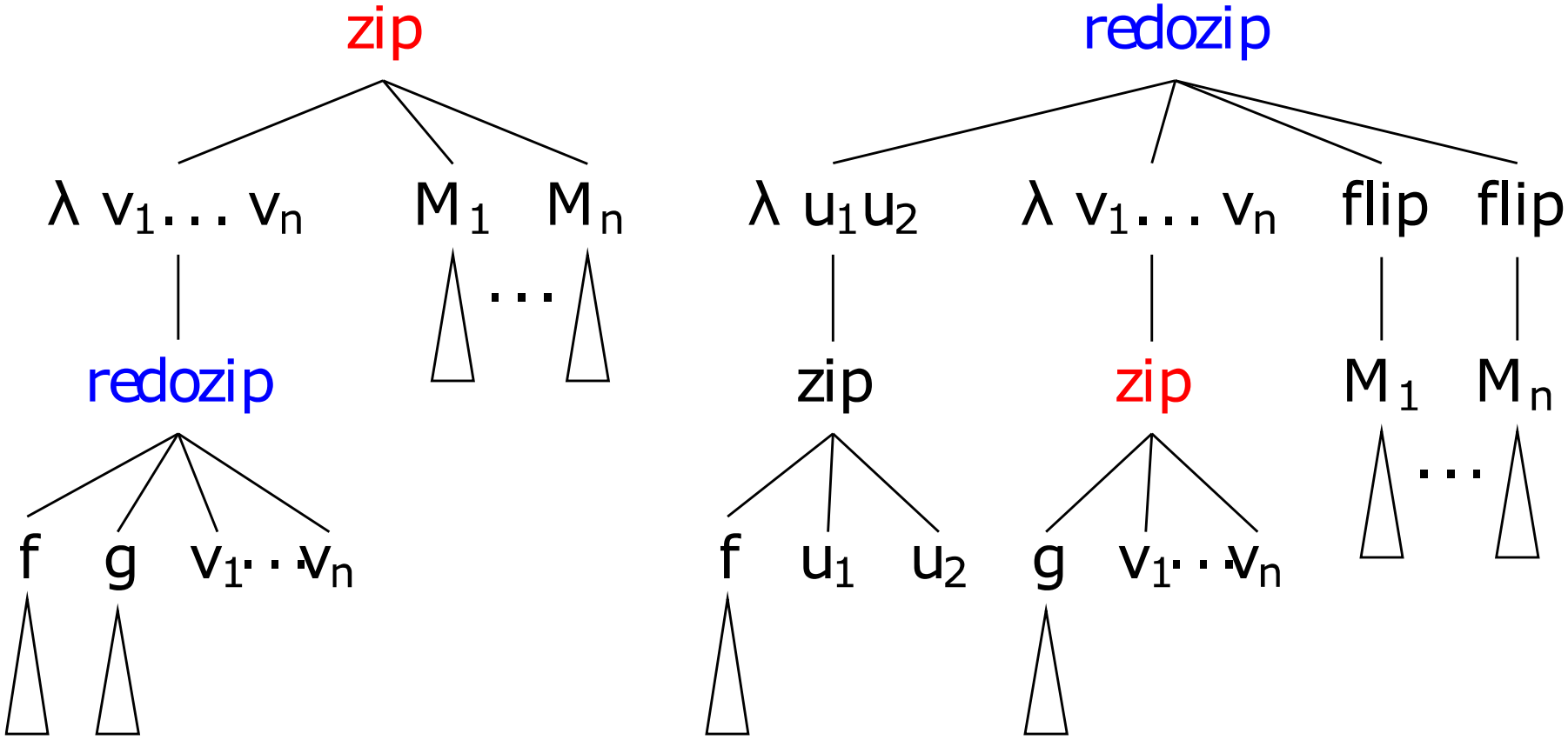



```
zip (\r -> redozip (+) (*) r u) rA
```

```
redozip (zip (+)) (\c q -> zip (*q) c) cA u)
```

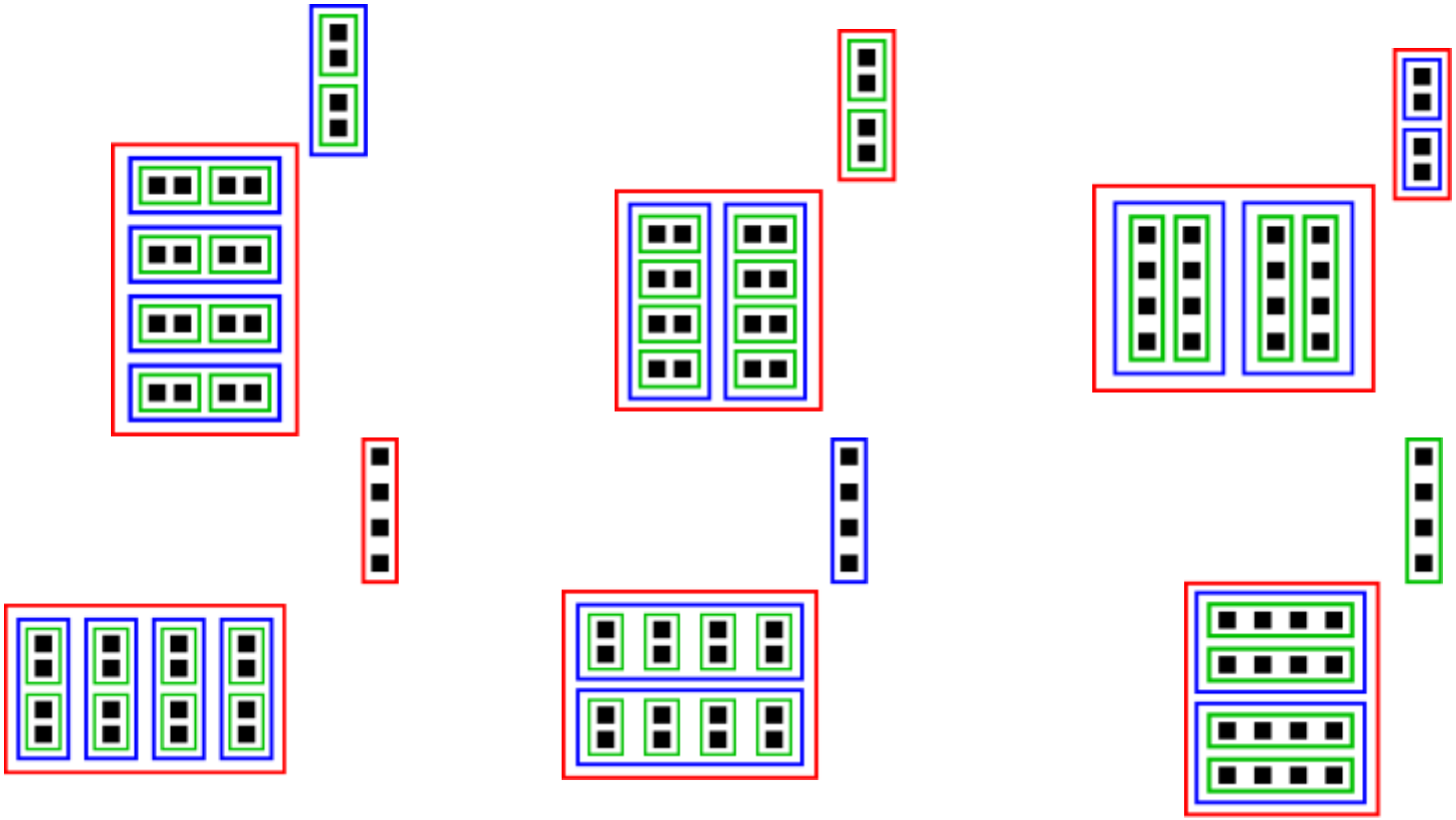
Same result, different performance

EXPRESSION TRANSFORMATIONS



EXPRESSION TRANSFORMATIONS

We can also subdivide operations and the logical layout of tensors:



Details:
Berényi, D, Leitereg, A, Lehel, G.
Towards scalable pattern-based optimization for dense linear algebra.
Concurrency Computat Pract Exper. 2018; 30:e4696. https://doi.org/10.1002/cpe.4696

EXPRESSION TRANSFORMATIONS

```
map (\r_A →
    map (\c_B →
        redozip (+) (*) r_A c_B) B) A
```

How does the performance change if we reorder?

naive

Operation ordering			Time [s]
mapA	redozip	mapB	0.45
redozip	mapA	mapB	1.41
mapA	mapB	redozip	4.67
mapB	mapA	redozip	6.05
redozip	mapB	mapA	13.8
mapB	redozip	mapA	15.6

But we need to compile and benchmark n! orderings…

PREDICTING PERFORMANCE

- Tensor contractions
  - Product of tensors in sums
  - $R_i = \sum_j A_{ji}$
  - $R_{ij} = \sum_k A_{ik} B_{kj}$
  - $R_{ij} = \sum_k A_{ikj} (\sum_l B_{li} C_{kl})$

- Run on a single core of an Intel® Xeon® E5-2650
- Measure with Google's benchmark library

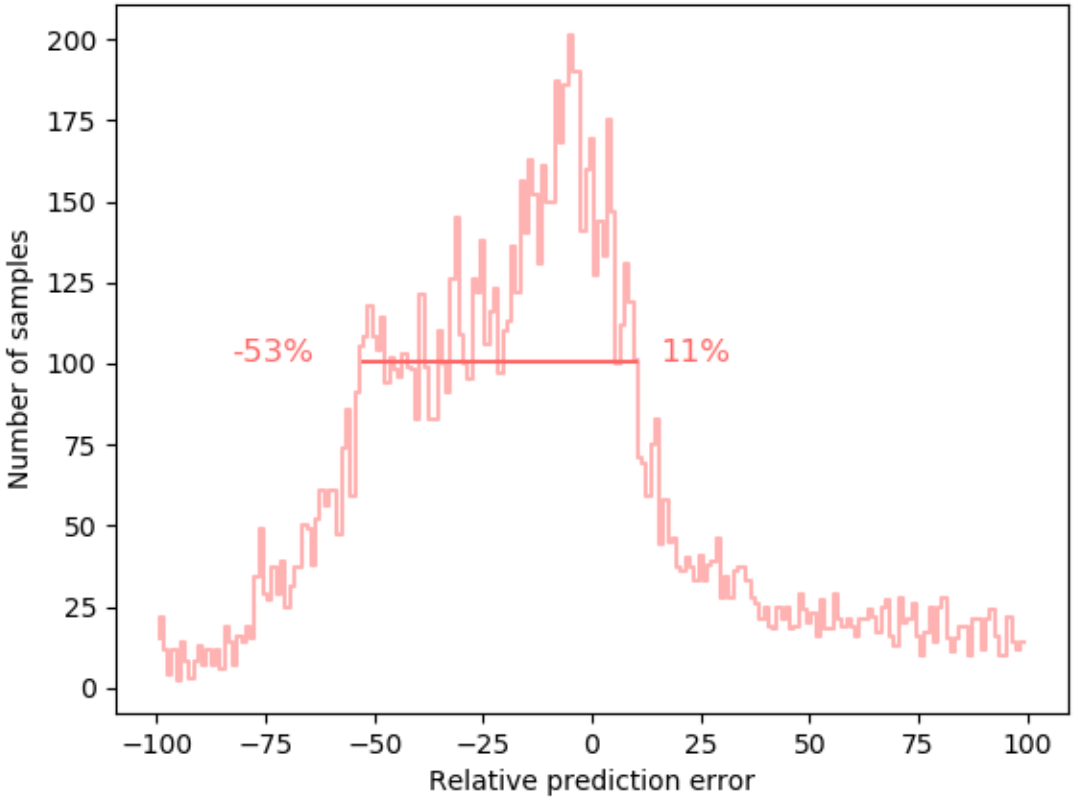
THE DATA SET

- Random contraction expressions with random dimension sizes
  - 1 or 2 sums
  - 1-6 tensors
  - 1-4 dimensions
- Modified matrix multiplications
  - $R_{ij} \sum_k A_{ki} B_{kj}$
  - $R_{ij} \sum_k A_{ijk} B_{kj}$
- 92k measurements, 63k with only 1 sum

A SIMPLE PERFORMANCE MODEL

- Linear combination of the number of
  - Arithmetic operations (+,∗)
  - Loads & stores
- Count these in a bottom-up traversal
- Find the best coefficients with linear regression

A SIMPLE PERFORMANCE MODEL



- Average error: 37%
- FWHM: 64%
- Samples within 5% error: 1636 (out of 12500)

PREDICTION ERRORS

```
// double A[388][2573];
// double R[2573];
for(int i = 0; i < 2573; ++i) {
    double sum = 0;
    for(int j = 0; j < 388; ++j) {
        sum += A[j][i];
    }
    R[i] = sum;
}
```

1.1 ms

```
// double A[388][2570];
// double R[2570];
for(int i = 0; i < 2570; ++i) {
    double sum = 0;
    for(int j = 0; j < 388; ++j) {
        sum += A[j][i];
    }
    R[i] = sum;
}
```

10.9 ms

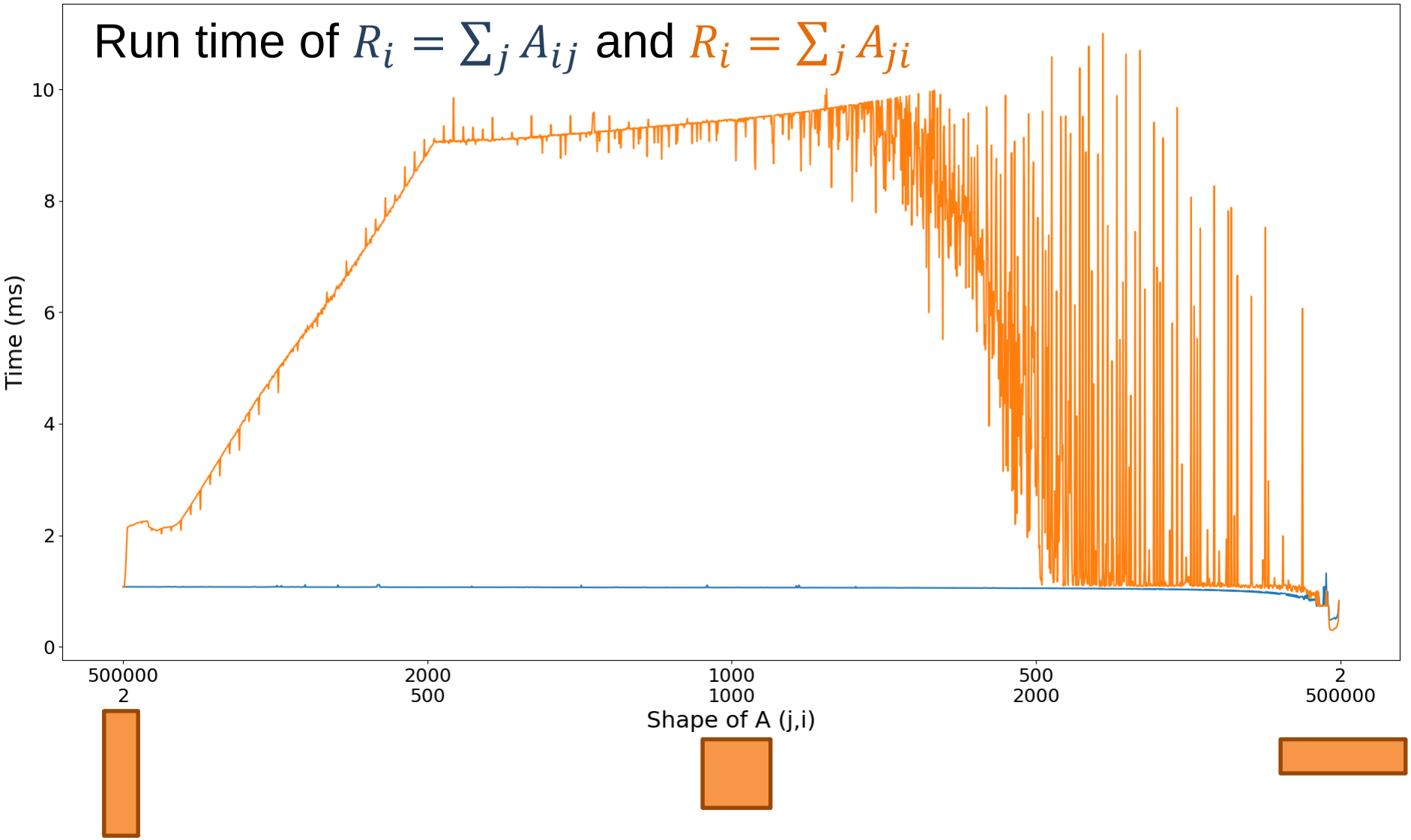
DATA LOCALITY

- Hierarchical cache (L1, L2, L3)
- Load operation time depends on cache level
- No direct way to control cache usage, we have to guess what happens

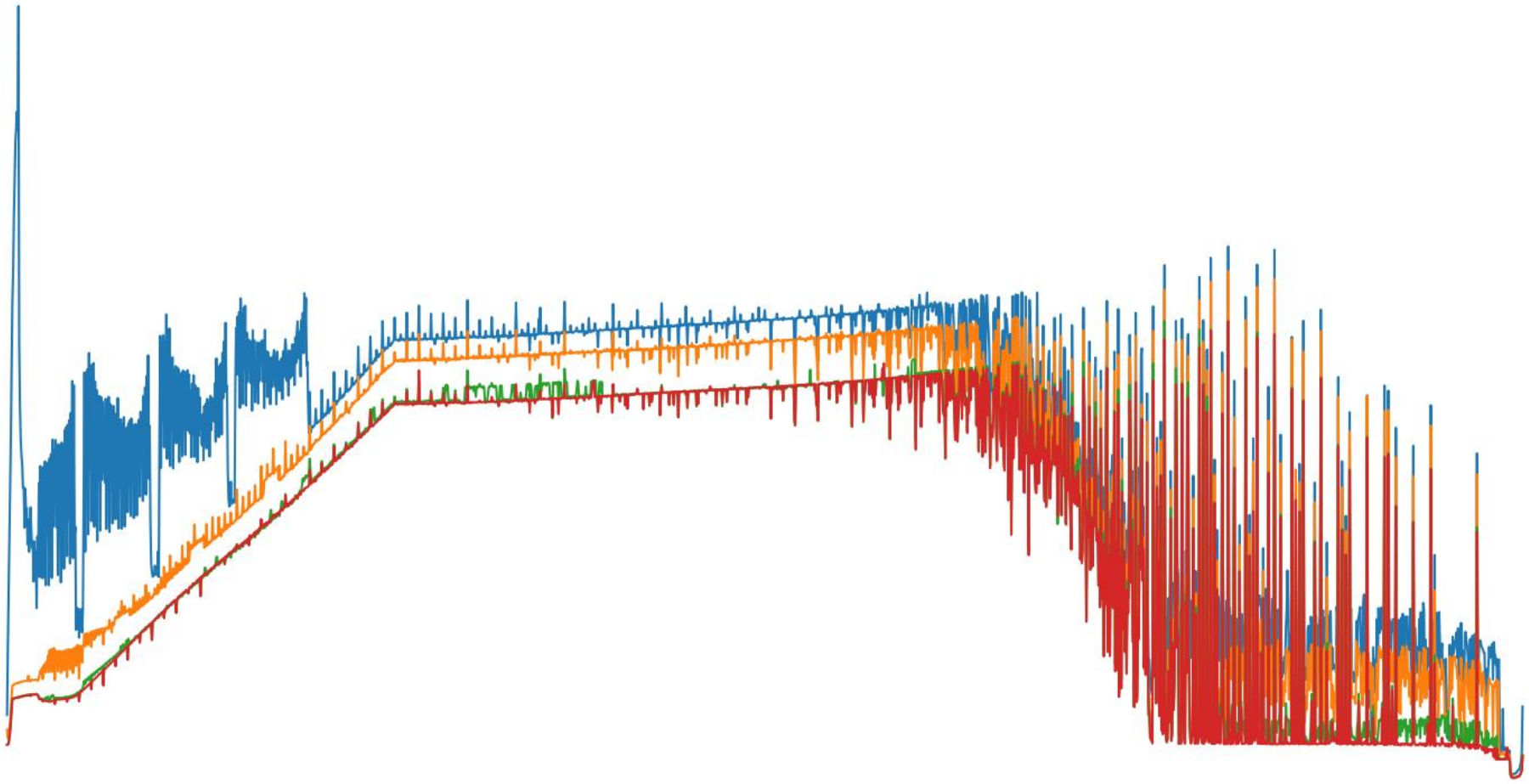
DATA LOCALITY

- A single read loads 8 elements (doubles) to a cache line
  - 2^n sizes can cause problems 
- Reading those at the next iterations: “free”
- Reading those later: it depends
  - On cache size
  - On cache policy 
  - On the sizes of dimensions
- And there is also prefetching 

PREDICTING PERFORMANCE



CPU AFFINITY



NEURAL NETWORK

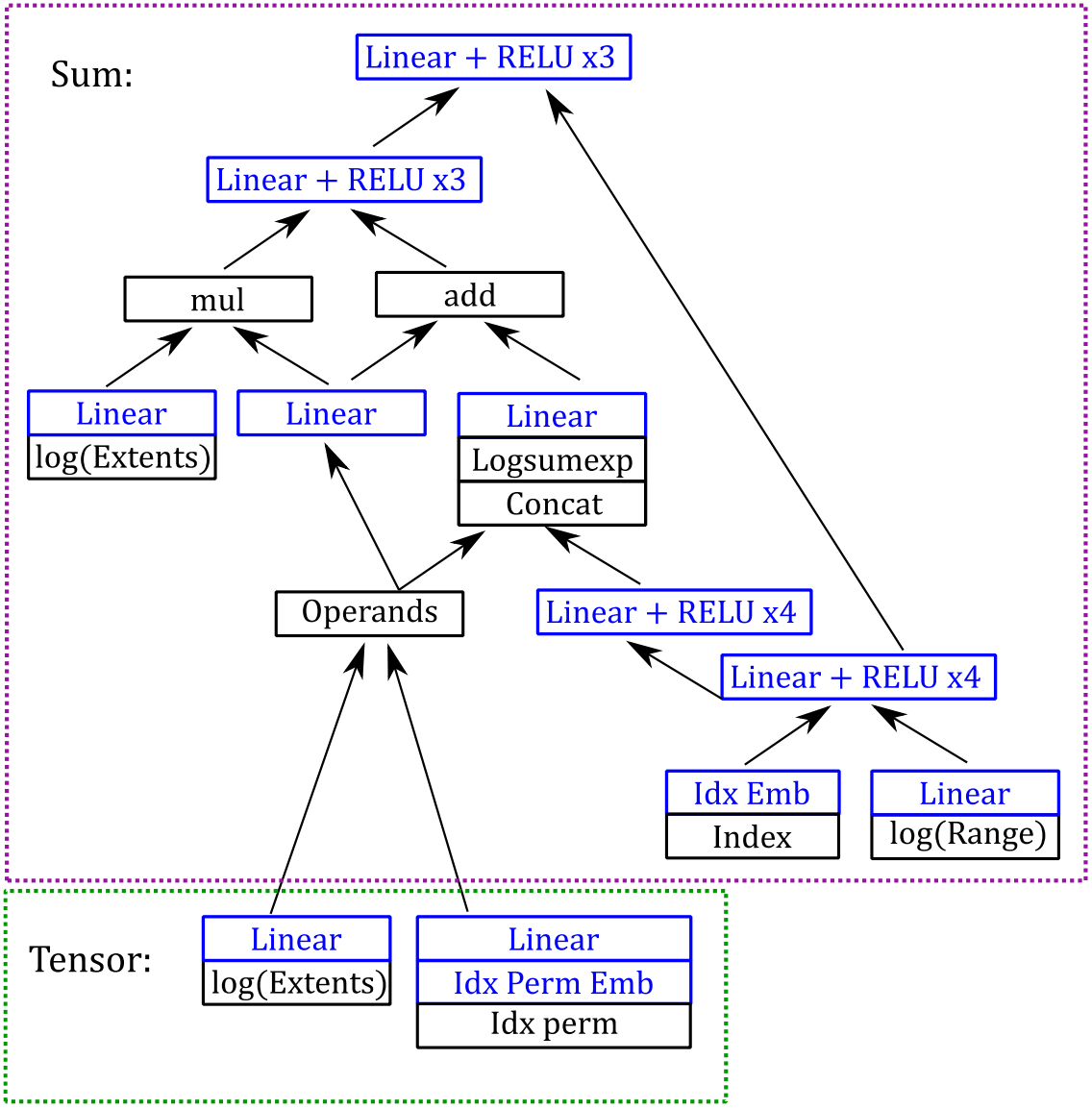
How to feed a tree into a neural network?

$$C_{ij} = \sum_k A_{ik} B_{kj}$$

- Index permutation embeddings
- Extents
- Range
- Operands

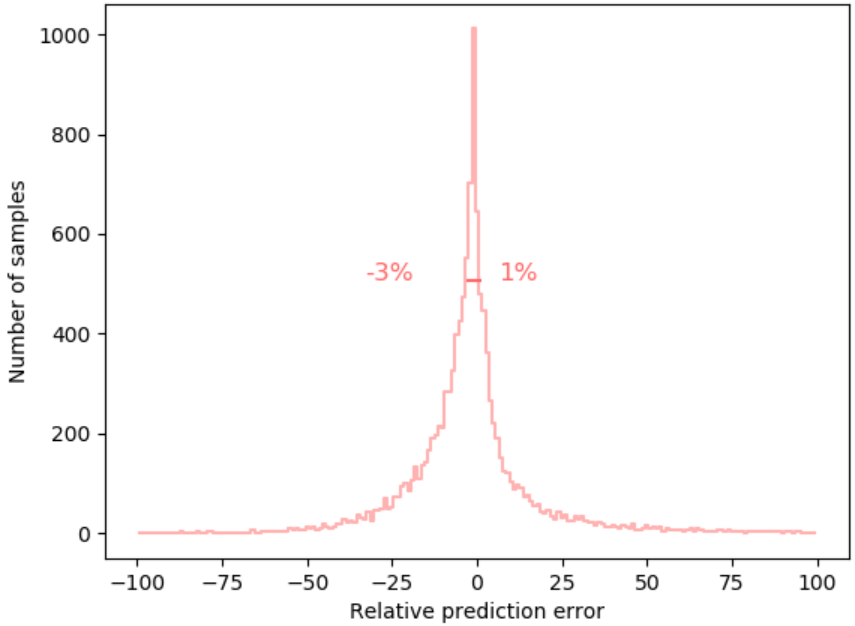
- Index permutation embeddings
- Extents

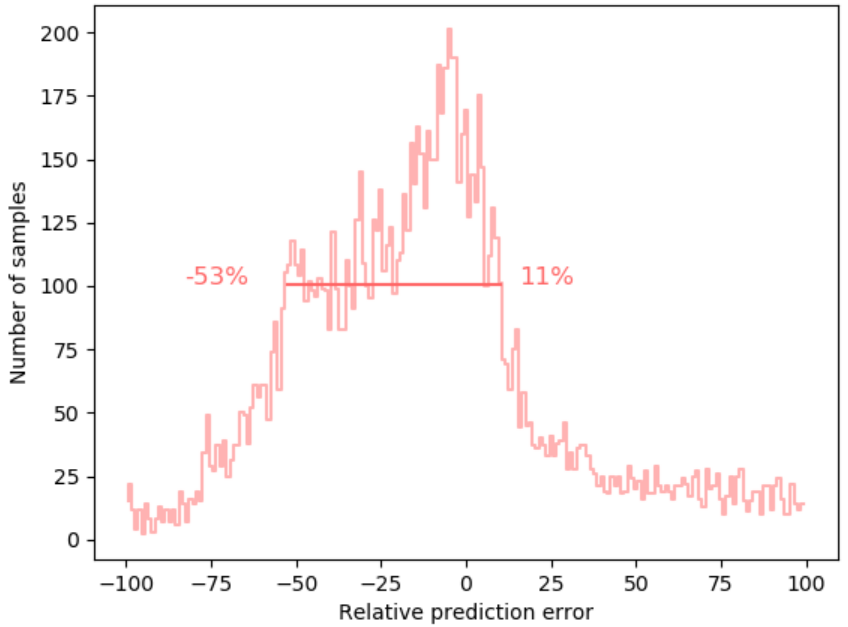
NEURAL NETWORK



- Blue boxes are parameters
- Linear (affin) transforms align the state vectors
- Additive and multiplicative component separated

RESULTS





13%	Average error	37%
4%	FWHM	64%
5361	Samples within 5% error (out of 12500)	1636

CONCLUSION

- Data locality heavily impacts performance
- But it is hard to model or predict
- Neural networks seem to grasp some effects

- This was only the beginning
  - The training data could be more representative
  - Predict relation, not time (enough for optimization)

THANK YOU FOR YOUR ATTENTION!

The LambdaGen project:
https://github.com/leanil/LambdaGen

We thank Róbert Csordás and Gábor Lehel for valuable discussions.