Programming abstractions and optimizing compilers for energy-efficient computing

Jeronimo Castrillon
Chair for Compiler Construction (CCC)
TU Dresden, Germany

1st Workshop on NetZero Carbon Computing (NetZero’23)
Montreal, Canada
February 25, 2023
Programming challenges

The **golden era** in computer architecture: Requires **next-gen programming methodologies** to **democratize** heterogeneous and emerging computing across domains.

In this talk: Examples for energy-efficiency & (a bit of) life-time

https://www.hpcwire.com/2017/04/10/nvidia-responds-google-tpu-benchmarking/
Semantic gap \(\rightarrow\) performance gap

\[ v_{ijk,e} = \sum_{i'=0}^{p} \sum_{j'=0}^{p} \sum_{k'=0}^{p} A_{kk'} A_{jj'} A_{ii'} u_{i'j'k'} e \]

What we want

What we (naively) code

What performance experts code

```c
void cfd_kernel()
{
    double A[restrict 7][7],
            u[restrict 216][7][7][7],
            v[restrict 216][7][7][7][7];

    /* element loop: */
    #pragma omp for
    for (int e = 0; e < 216; e++)
    {
        double t6[7][7][7];

        /* 1st contraction: */
        #pragma simd
        for (int i0 = 0; i0 < 7; i0++)
        {
            double t8 = 0.0;
            for (int i3 = 0; i3 < 7; i3++)
                t8 += A[i0][i3] * u[e][i1][i2][i3];
            t6[i0][i1][i2] = t8;
        }
        /* end of 1st contraction */
        #pragma simd
        for (int i4 = 0; i4 < 7; i4++)
        {
            double t9 = 0.0;
            for (int i7 = 0; i7 < 7; i7++)
                t9 += A[i4][i7] * t6[i5][i6][i7];
            t7[i4][i5][i6] = t9;
        }
        /* end of 2nd contraction */
        #pragma simd
        for (int i8 = 0; i8 < 7; i8++)
        {
            double t10 = 0.0;
            for (int i11 = 0; i11 < 7; i11++)
                t10 += A[i8][i11] * t7[i9][i10][i11];
            v[e][i8][i9][i10] = t10;
        }
        /* end of 3rd contraction */
    }
    /* end of element loop */
}
```
Semantic gap $\Rightarrow$ performance gap

$v_{ijk,e} = \sum_{i'=0}^{p} \sum_{j'=0}^{p} \sum_{k'=0}^{p} A_{kk'} A_{jj'} A_{ii'} u_{i' j' k'} e$

What we want

What we (naively) code

What performance experts code

AI accelerator

https://www.hpcwire.com/2017/04/10/nvidia-responds-google-tpu-benchmarking/

Example 1: 5G baseband processing

- E4C collaborative project: Extreme energy-efficient cloud radio access network
- Domain-specific modeling of comm. workload
- Novel server architecture (links, PCB, …)

Domain-specific knowledge to scale up

- Dynamic dataflow model of computation
- Domain-specific global optimization
- Traffic models for different scenarios

Khasanov, ACM TECS 2021
Domain-specific knowledge to scale up

- Dynamic dataflow model of computation
- Domain-specific global optimization
- Traffic models for different scenarios

Germany-wide what-if analysis: \( \sim 400 \text{ ktC}_2\text{eq} \)
First place: BMBF competition in green ICT platform

https://bit.ly/3LXdskSo
Example 2: HPC physics simulation

- Domain-specific languages (DSLs)
  - Code closer to what we want
  - Suitable abstractions ➔ powerful optimization

- MLIR-based compilation

\[
t = (S \otimes (S \otimes (S \otimes u))_{cz}^{xyz})_{by}^{cxy} \}_{ax}^{bca}
\]

What we want

\[
v_{ijk,e} = \sum_{i'=0}^{p} \sum_{j'=0}^{p} \sum_{k'=0}^{p} A_{kk'} A_{jj'} A_{ii'} u_{i'j'k'e}
\]

source = ...
var input A : matrix &
var input u : tensorIN &
var input output v : tensorOUT &
var input alpha : [] &
var input beta : [] &
\textbf{v} = \textbf{alpha} * (A \# A \# A \# u) . 
[[5 8] [3 7] [1 6]]) + \textbf{beta} * \textbf{v}

Rink, RDWSL’18
Susungi, GPCE’18
Rink, Array’19
H2020 EU Project: Convergence HPC, Big Data and ML
Transformations for a \textbf{17x speedup} (same precision)

S. Soldavini, ACM TRETS'22

https://everest-h2020.eu
FPGA code generation: HBM FPGA

- H2020 EU Project: Convergence HPC, Big Data and ML
- Variants with up to **24x better energy efficiency**

Goal: Transparent to domain-expert (user of e.g., WRF framework)
Example 3: Emerging data-centric architectures

- Compute in-place, avoid data movement, transformations to match primitives
- Novel architectures for near-memory and in-memory computing

Samsung, Lee, Sukhan, et al. ISCA 2021
End-to-end compiler for PCM acceleration

- MLIR frontend for general tensor expressions
  - Reuse GEMM transformations from \texttt{linalg} (in MLIR)
  - Transformations for 
    \textbf{performance} and \textbf{wear-leveling}

\[ v = (x, y) \]

The Open CIM Compiler (OCC)

1. GEMM-centric hardware-agnostic passes
2. hardware-specific passes

A. Siemieniuk, IEEE TCAD 2021
Optimization results: Beyond Matmult
Generalized MLIR infrastructure

- Entry: linear algebra abstraction (common to ML frameworks and beyond)
- Intermediate languages for in and near memory computing
- Target-specific models and optimizations

A. Khan, Arxiv, 2023
Hyper dimensional computing (HDC)

- HDC: Embed data in 10 k-dimensions – Von-Neuman Bottleneck!
- Leverage bulk-wise binary operations

1. Input Items
2. Mapping to HD-Space
3. Base Vectors
4. Mapping to HD-Space
5. Base Hypervectors
6. HD Vector Rotation
7. Permutation and Binding
8. Bundling
9. Similarity Check
10. Text Corpus
11. Training Data
12. Population Count
13. Result: Closest Similarity

Khan, A. A., et al. ACM TECS 2022
Hyper dimensional computing (HDC)

~6x faster and 5.3x less energy over FPGA accelerator

Khan, A. A., et al. ACM TECS 2022
Challenging & exciting computing landscape!

Tools automation: Need better understanding of full-life-cycle sustainability aspects, beyond energy-efficiency
Thanks! & Acknowledgements

..., and previous members of the group (Andres Goens, Norman Rink, Sven Karol, Sebastian Ertel), and collaborators (J. Fröhlich, I. Sbalzarini, Alex K. Jones, A. Cohen, T. Grosser, T. Hoefler, H. Härtig, H. Corporaal, C. Pilato, S. Parkin, P. Jääskeläinen, K-H Chen)