From models to ML and back again

Michael O’Boyle
Lemma 1 Given $T'_{\ell-1}$ such that $LT'_{\ell-1}, UT'_{\ell-1}$ are jointly invariant for $j \in i_1, \ldots, i_{\ell-1}$ i.e.

\[
LT'_{\ell-1} e_{ik} = UT'_{\ell-1} e_{ik} = e_{ik} \forall k \in 1, \ldots, \ell - 1 \tag{36}
\]
\[
e_{ik}^T LT'_{\ell-1} = e_{ik}^T UT'_{\ell-1} = e_{ik}^T \forall k \in 1, \ldots, \ell - 1 \tag{37}
\]

Then $T'_{\ell-1}$ can be chosen so that:

\[
T'_{\ell-1} e_{ik} = e_{ik} \forall k \geq \ell \tag{38}
\]
Why do compilers fail?

Wanted to see if we could improve performance
- Huge gap between compiler and library/programmer

We looked at matrix multiplication
- World’s most studied compiler problem
- 2 transformations
- See how good existing approaches were

What does the space look like?
- Can knowledge help build better compilers
Main conclusions

• Space is hard
  – especially if hardware changes

• All Existing Compiler Analysis
  – FAILED
  – most studied benchmark

• Need empirical evidence
  – rather than theory

• A massive cultural shift

• 2 ways forward
  – Search
  – Machine Learning
Why not use knowledge of other programs?

New Program

Simplest form of machine learning
Prior knowledge speeds up
Compiler writer model
Compiler writer model

Fiction

Reality
• Given enough data and right model
  – Can automatically capture behaviour
  – Can predict outcomes
  – Without knowing internal
Lessons learnt

- Hand crafted optimisation
  - doomed to fail
  - based on fiction
- Let go of certaint
  - dynamic analysis
  - statistical modelling
  - automatic prediction via ml
- Hardware is probabilistic
  - why not software?
Traditional approach
Traditional approach

Appeal to reason
Traditional approach

- Appeal to reason
- Express problem: set of equations - simple terms
- Develop an elegant model by hand
Traditional approach

Appeal to reason
Express problem: set of equations - simple terms
Develop an elegant model by hand
Find some benchmarks that fit
Traditional approach

- Appeal to reason
- Express problem: set of equations - simple terms
- Develop an elegant model by hand
- Find some benchmarks that fit
- Test your technique against your competitors
  Iterate over benchmark selection till you win!
Traditional approach

- Appeal to reason
- Express problem: set of equations - simple terms
- Develop an elegant model by hand
- Find some benchmarks that fit
- Test your technique against your competitors
  Iterate over benchmark selection till you win!
New approach
New approach

Select large benchmark set
Create a large optimization
New approach

Select large benchmark set
Create a large optimization

Analyse it - room for improvement?
If not, do something else!
New approach

Select large benchmark set
Create a large optimization

Analyse it - room for improvement ?
If not, do something else!

Only then model automatically
New approach

Select large benchmark set
Create a large optimization

Analyse it - room for improvement? If not, do something else!

Only then model automatically

Separate test set - no cheating!
New approach

Select large benchmark set
Create a large optimization

Analyse it - room for improvement?
If not, do something else!

Only then model automatically

Separate test set - no cheating!
Statistical modelling or machine learning is the cherry on the top. It’s all about the data.
Statistical modelling is the cherry (hammer) on the top
Evidence based compiler construction
Significant step in the maturation of a science
Data driven modelling

A tool looking for nails!
Nail 1: Auto-parallelization
State of art: No speedup

NAS PB
NAS NPB 2.3 OMP-C and SPEC CFP2000
2 Quad-cores (8 cores in total) Intel Xeon X5450 @ 3.00GHz
Intel icc 10.1 -O2 -xT -axT -ipo
Wide performance gap

Manual: 3.4x speedup
Wide performance gap

Manual: 3.4x speedup
Data driven approach
Data driven approach

Sequential code in C

Profiling-driven analysis

Code with OpenMP annotations

Machine-Learning based mapper

Code with profitable loops

Use data twice

[PLDI09]
Use profiling to gather evidence - not proof-of potential parallelism
Data driven approach

- Sequential code in C
- Profiling-driven analysis
- Code with OpenMP annotations
- Machine-Learning based mapper
- Code with profitable loops

Use prior knowledge to determine mapping
Data driven approach

Sequential code in C → Profiling-driven analysis → Code with OpenMP annotations → Machine-Learning based mapper → Code with profitable loops

Ask user to double-check

Predict if worthwhile
Our approach: 3.34 average speedup on 8 cores

NAS NPB 2.3 OMP-C and SPEC CFP2000
2 Quad-cores (8 cores in total) Intel Xeon X5450 @ 3.00GHz
Intel icc 10.1 -O2 -xT -axT -ipo
96% of hand-parallelized
Nail 2: Optimising GPUs
Comparison against hand-coded Portable OpenCL compiler
Portable OpenCL compiler

Can we use data driven approach to EXPLAIN behaviour? [SC13]
3 transformations
- Thread coarsening: using divergence analysis
- Stride optimisations
- Work group size

Can we use data driven approach to EXPLAIN behaviour? [SC13]
Coarsening helps: Matrix

Speedup vs Coarsening Factor

- Cypress
- Tahiti
- Core–i7
- Fermi
- Kepler
43,000 experiments

- 17 Benchmarks: Parboil and SDKs
- 5 Platforms
  - Nvidia: Fermi, Kepler
  - AMD: Radeon and Cypres
  - Intel CPU: Core i7

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Possible values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coarsening factor</td>
<td>{1, 2, 4, 8, 16, 32}</td>
</tr>
<tr>
<td>Coarsening dimension</td>
<td>{0, 1}</td>
</tr>
<tr>
<td>Stride</td>
<td>{1, 2, 4, 8, 16, 32}</td>
</tr>
<tr>
<td>Local work group size</td>
<td>{dev min, \ldots, 32, 64, 128, \ldots, dev max}}</td>
</tr>
</tbody>
</table>
Comparison against hand-coded
Hard to find: Fermi

Comparison against hand-coded

binarySearch blacksholes convolution dwtHaar1D fastWalsh floydWarshall mriQ mt mtLocal mvCoal mvUncoal nbody reduce sgemm sobel spmv stencil

Speedup

0.0 0.5 1.0 1.5 2.0 2.5
Hard to find: i7

Comparison against hand-coded

Core–i7

Speedup

binarySearch
blackholes
convolution
dwtHaar1D
fastWalsh
floydWarshall
mriQ
mt
mtLocal
mvCoal
mvUncoal
nbody
reduce
gemm
sobel
spmv
stencil

9.25
Best Factors: Kepler

Kepler

Stride

32
16
8
4
2
1

binarySearch blackholes convolution dwtHaar1D fastWalsh floydWarshall mriQ mLocal mvCoal mvUncoal nbody reduce sgemm sobel spmv stencil
Best Factor: Cypress

Cypress

Stride

32
16
8
4
2
1

binarySearch, blacksholes, convolution, dwtHaar1D, fastWalsh, floydWarshall, mriQ, mt, mtLocal, mvCoal, mvUncoal, nbody, reduce, sgemm, sobel, spmv, stencil
Analysis: Fermi

```
loads < 0.92
Y
N
branches < 2.8
```

```
1.70
```

```
CacheMisses < 1.00
```

```
1.06
0.81
```

0.4
Analysis: Cypress

ALUPacking < 1.28
  Y
  0.8
  N
  ALUBusy < 0.59
    0.79
    2.10
Analysis: i7  Loads unimportant

```
vectorInsts < 1.42
  Y
  branches < 0.7
    Y
    ipc < 1
      1.13
    1.47
  3.8
```


Nail 3: Runtime Scheduling with external work loads
Assumption

Target = Our program
Target = Our program
Environment = Workload + Hardware resources
Workload and hardware vary

Compiler - need to consider environment
  - blur with runtime
Small experiment

*LU* co-scheduled with workloads

- **Target** workload: \( W_{1,8} \)
- **Workload**:
  - \( W_{1,8} \)
  - \( W_{2,6} \)

**Graph:**
- Y-axis: Speedup
- X-axis: Algorithms
- Default, Best Static, Hill Climbing, Oracle

**Bar Chart:**
- Large gap > 2x
Small experiment

LU co-scheduled with workloads

Can we use data and ml to build a better model?
- combine static code features and dynamic information
Training
- vary targets and workloads
- separate training and test
Collect data + learn a model

Training
- vary targets and workloads
- separate training and test

1.52x over hill climbing
[CGO13]
What happens if scenario

Fundamental issue with training
What happens if scenario

Fundamental issue with training

Answer: Predict environment
Environment prediction

- Actual environment features
- Predicted environment features
- Predicted # threads
- Adjusted # threads

The diagram illustrates the relationship between environment features and the number of threads, showing how predictions are made and adjustments are applied based on the difference between actual and predicted values.
Mixture of experts

Online Expert Selector $M$

Expert 1

Expert 2

Expert $k$

input $f = [c, e]$

output $n_{\text{best}}$

$n_1$

$n_2$

$n_k$

$\hat{e}_1$

$\hat{e}_2$

$\hat{e}_k$

offline experts

Expert selected based on accuracy of environment prediction
Mixture of experts

Outperforms all existing analytic/ml approaches [PLDI’15]
Benefits of data-driven
Benefits of data-driven

- Evidence based
- Oracle study
- Beating random
- Separate training and test
Benefits of data-driven

- Evidence based
- Oracle study
- Beating random
- Separate training and test
- Allows natural automation
Lessons Learnt 1

- ML is highly successful for profitability analysis
  - Model often irrelevant - it’s all about the data
- Should be de facto approach.
  - Analytic models should be viewed as priors
- Community still skeptical
  - Lack of control
  - Black box magical thinking
  - Signal/Noise
  - Correlation not equal to causality
Lessons Learnt

• Easier to get away with a wrong explanation than a correct prediction!
  • Features key - deep learning as an alternative
• Real Problem-
  • Not enough data
• When data is plentiful, simulators, compiler flags
  • less interest!
• Good at predicting within a design space, not new designs
  • Transfer learning key
• Newer ML yet to be exploited
  • Auto encoder, shared layers in Boltzmann machines etc
What next
What next

Endless automation!
What next

Endless automation!

I want to map any code to any new heterogeneous hardware platform with minimal human intervention and minimal a priori knowledge.
What next

Endless automation!

I want to map any code to any new heterogeneous hardware platform with minimal human intervention and minimal a priori knowledge.

Use different machinery - constraint language, program synthesis.

Back to models!
Libraries and DSLs are the new het. API/ISA

Detect code structures (idioms) that match het APIs

Idioms:
- Dense linear algebra,
- Sparse Mv,
- Stencils,
- Reductions
- Histograms

APIs:
- cuBLAS, clBLAS
- cuSparse, clSparse
- Halide, Lift

Platform
- AMD APU: multicore (+Radeon) (+NVIDIA Titan)
Finding Idioms using Constraints

Program

- SAT solver
- Translate to DSL
- Stitch together

Constraints

Language Def

LLVM
Auto-discovery: Synthesis + Induction
Auto-discovery: Synthesis + Induction

Interrogate + Synthesise
Auto-discovery: Synthesis + Induction

Interrogate + Synthesise

Transform
Auto-discovery: Synthesis + Induction

Interrogate + Synthesise  →  Transform  →  Induction  →  IDL + Map
Auto-discovery: Status

```c
int a, b, c:
  c = a + b

int a[8], b[8], c:
  c += a[] * b[]

int n, a[n], b[n], c[n]:
  c[] = a[] * b[]

BLAS1,2,3, Sparse, Crypto

Induction

Idioms, DSLs, General Languages
```
Connections

- Program Synthesis is a search space technique
- Iterative compilation is search based

- Can model (certain) hardware as black box
- Use ML to model

- ML and Program synthesis connected
- Generalising examples related to Inductive Logic Programming

- Dancing around black hole of undecidability

- Long term goal
  - Given signature(…syntax) of hardware API (….prog language)
  - Can we automatically map from code to it using examples (…semantics)
Legacy Program

Adapting to Change
Legacy Program

... much later
Adapting to Change

Legacy Program

... much much later
• Hand-crafted optimization
  – doomed to fail
• ML, data-driven optimisation
  – dynamic analysis
  – statistical modelling
• Managing heterogeneous manycore
  – the challenge for a decade to come
• Want to learn about
  – Program Synthesis
  – Program Lifting
  – Inductive Logic Programmiing
  – Connection to ML
  – Decidability limits