Machine Learning for automating compiler/architecture co-design

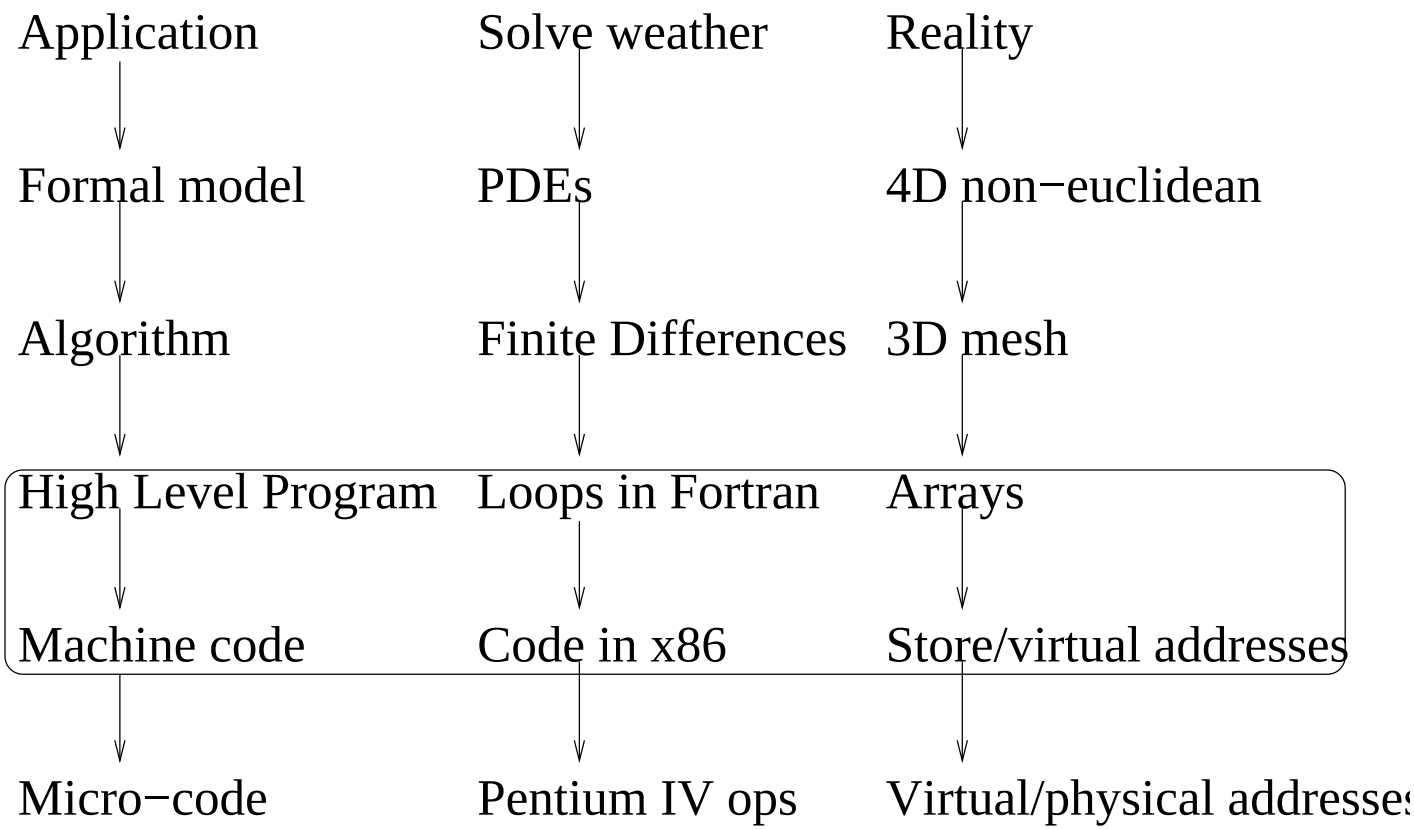
Michael O'Boyle

Institute of Computer Systems Architecture

School of Informatics
University of Edinburgh
UK

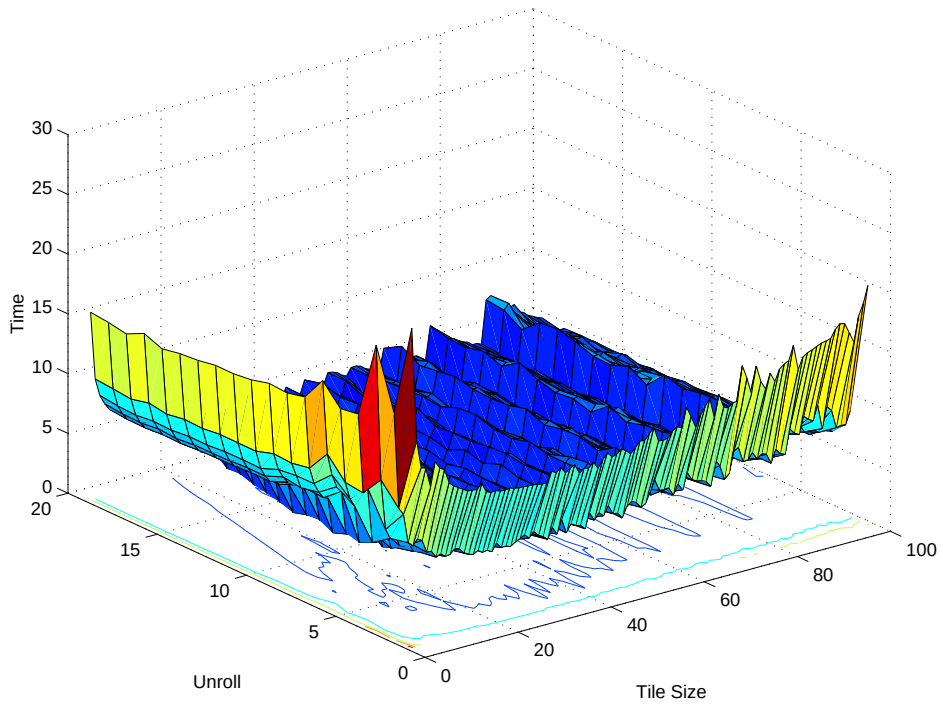
October, 2008

Layman's terms: Compilation:Linking users with hardware



- The glue between application code and hardware
  - Enable abstraction from ever-changing hardware
- Focus on performance - an optimisation problem
  - Area of study for 50 years. Why?

Why do compilers fail to find the best optimisation?



- Fundamental reason for failure is complexity and change
  - Architecture behaviour so complex - impossible to determine the best code sequence *a priori*.
  - Arch changes means we're always playing catch up
  - Standard approaches fail - so let's use ML!

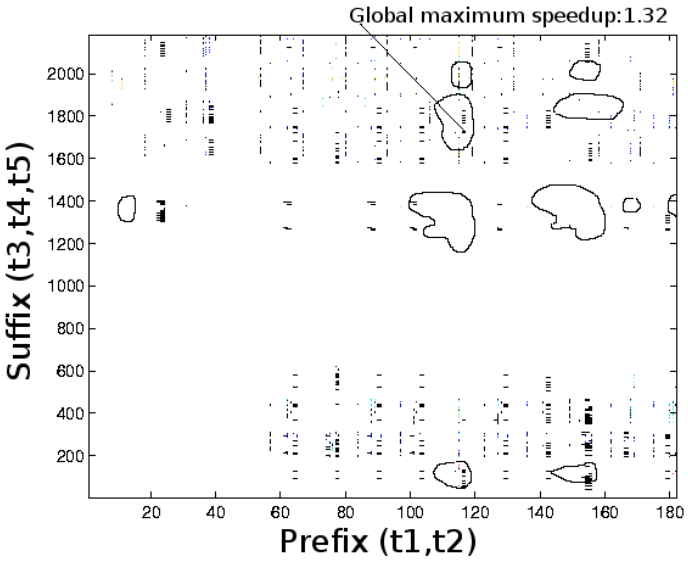
Automated compiler/architecture co-design

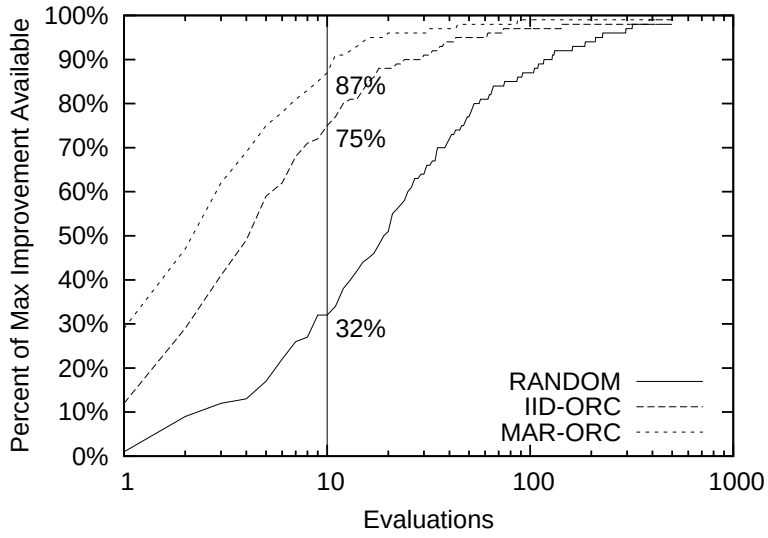
- Computer Architecture is concerned with design/layout of processor
  - A massive design space: num registers, size of caches etc.
  - Mutli-layered: ISA, micro-arch, verilog, netlist, polygons
- Architecture strongly dependent on compiler performance
- Ideally: predict performance of a yet to be built optimising compiler on a yet to be built architecture
  - Across massive design space
- Challenging
  - Again - Let's use ML

Problems stated in ML terms

- Predict best transf, sequence $\mathbf{P} \mapsto \mathbf{s}$ or $(\mathbf{P}, \mathbf{a}) \mapsto \mathbf{s}$
  - Input: Program Features $\mathbf{P}$, sequence $\mathbf{s}$, Output time t
  - For each program $\mathbf{P}$ build distribution over good sequences $q(\mathbf{s}|\mathbf{P})$ based on output, time t
  - For new program $\mathbf{P}^*$ determine nearest neighbour in feature space and use its distribution.
  - Either max value for one-shot prediction or whole distribution to bias search.
- Predict time of new program on architecture space $(P, \mathbf{a}) \mapsto \mathbf{t}$
  - Input: Architecture Features $\mathbf{a}$, Output time t
  - For each program P build model (ANN) $P, \mathbf{a} \mapsto \mathbf{t}$
  - For new program P^* evaluate a few $\mathbf{a}^*$, t^* pairs
  - Express $P^*, \mathbf{a}^* \mapsto t^*$ as linear combination of prior $P, \mathbf{a} \mapsto \mathbf{t}$.
  - Use this to predict new prog space. Uses outputs as features.

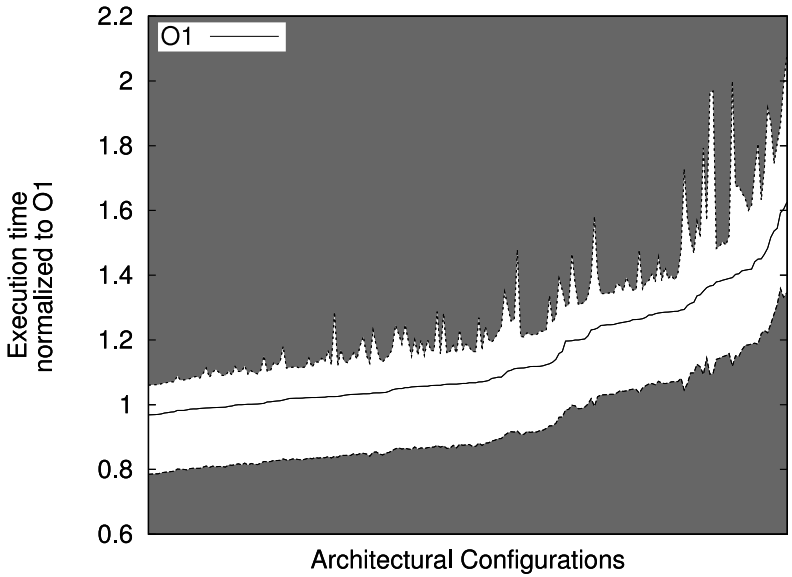
What have you done?: Focused search

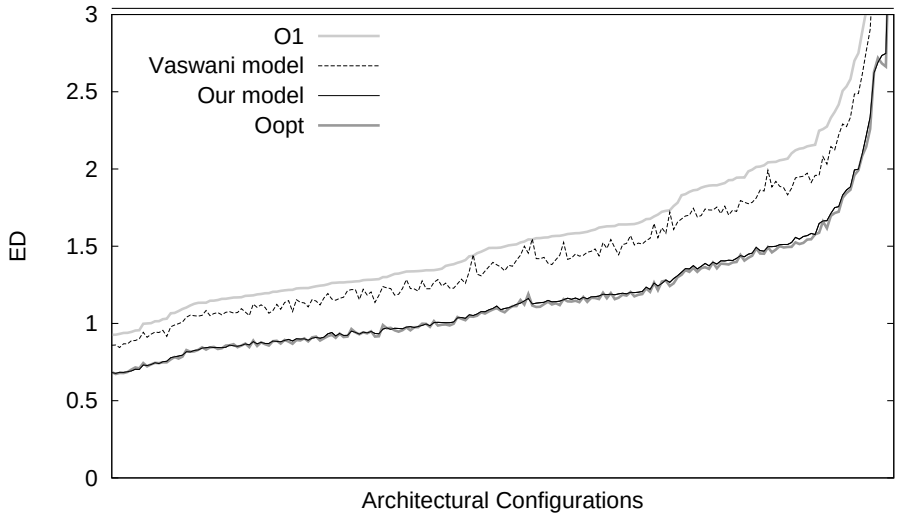




- Used nearest neighbour + distribution summary models to focus search
  - IID $P(s_1, s_2, \ldots, s_L) = \prod_{i=1}^{L} P(s_i)$.
  - Markov: $P(s_1, s_2, \ldots, s_L) = P(s_1) \prod_{i=2}^{L} P(s_i|s_{i-1})$.
- Feature space based on program syntax + PCA

What have you done?: Predicting co-design space





- Arch space
  - Used linear comb of $P, \mathbf{a} \mapsto \mathbf{t}$ for a new prog P^*
- Co-design
  - Used SVM and output(perf counters) of one run.
  - No transference. $\mathbf{C}, \mathbf{a} \mapsto \mathbf{t_{opt}}$
- Accurately predict the performance of an optimising compiler
  - Without having to build it!

Experience in ML

- Models
  - Linear/logistic regression, ANN, SVM, Gaussian Processes
  - GA and GP
  - IID and Markov
- Features
  - Automatic feature generation and selection - searching the feature space
  - Mutual information,
  - Outputs as features (responses)
  - Unsupervised learning for clustering
- Sequences and time
  - Markov model
  - Reinforcement learning
  - Combined off-line and on-line learning

Problems faced so far

- The main problem until recently has been generating data
  - One point out of a 18 billion design space = 1 week
  - Still difficult - multi-proc simulators
- Truly massive design spaces:
  - Multi-dimensions, many interlocking layers, combinatorial sub-problems
- Transference is difficult
  - Within transformation/micro-arch space ok
  - Across archs, across programs hard
- Input Data is complex tree/graph structure.
  - Eventually want to learn a model $\equiv$ program (ILP)
- Often non-Gaussian noise
  - Driving some to simulation
  - Also typically multiple sub-layers are modelled with error
- Sceptical community
  - Have to show worth while, have to beat all other techniques and explain how it all works

How widely used/important is ML used in your area?

- Potted History
- Compiler World
  - We started on search in 1997 OCEANS project
  - Cavazos and Moss looked at predictive modelling NIPS 1997
  - By 2002 old hat, by 2005 great new idea!
- Architecture World
  - More used to ML - perceptron based Branch predictors
  - Focused on speeding up simulation
  - Then predicting performance
  - No search or predicting best
- Really hot topic now
  - MilePost GCC: IBM support
  - EU and DARPA initiatives
  - Companies starting to take note
  - Will be a standard tool in a decade

What do you hope to gain from an ML research program?

- Advertise that ML is revolutionising system architecture design
  - Majority of ICSA use it in some way
- Get exposure to other ways of modelling problems
  - Our experience - ML technique not always critical
  - Features matter as does asking the right question
- ML expertise from Chris and Edwin has been critical to our success
  - Would like to continue success with deeper knowledge
- Get insight into why some areas are more ML friendly
  - IPAB and ICCS seem part of ML community
- Raise profile and lobby for Systems/ML appointments