Crowdsourcing auto-tuning

challenges and possible solutions

Many thanks to Domingo Giménez, Takeshi Iwashita, Reji Suda
for iWAPT organization and invitation

Grigori Fursin
INRIA, France

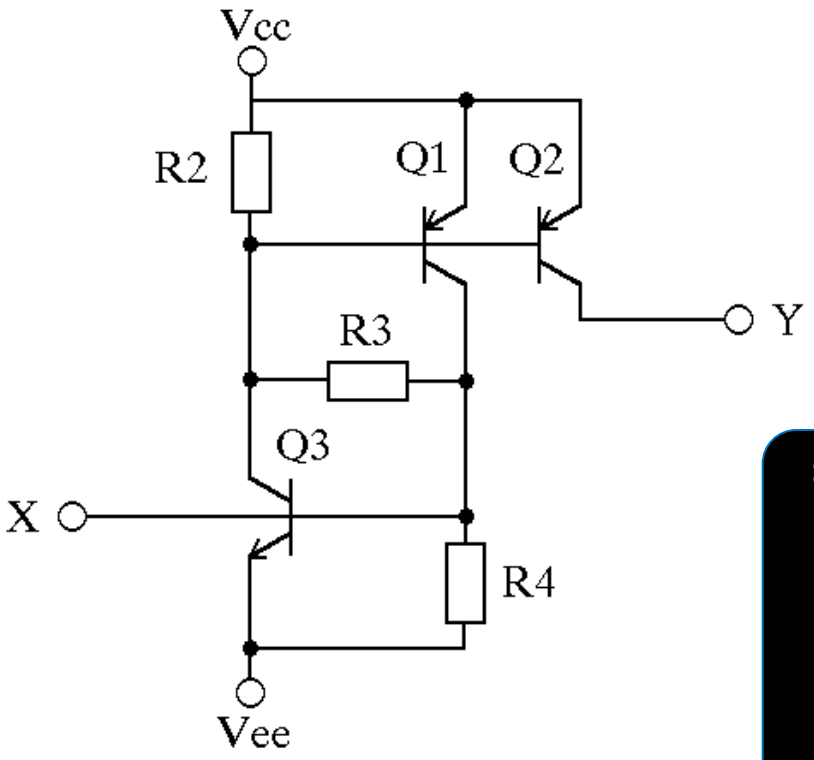
iWAPT 2013 @ ICCS 2013
June 2013, Barcelona, Spain

Message

- Revisiting current computer design and optimization methodology
- Leveraging experience and computer resources of multiple users
- Systematizing auto-tuning, predictive modelling and data mining to improve computer systems
- Starting international initiative to build collaborative R&D infrastructure and public repository of knowledge
(EU HiPEAC, USA OCCAM, various universities and companies)

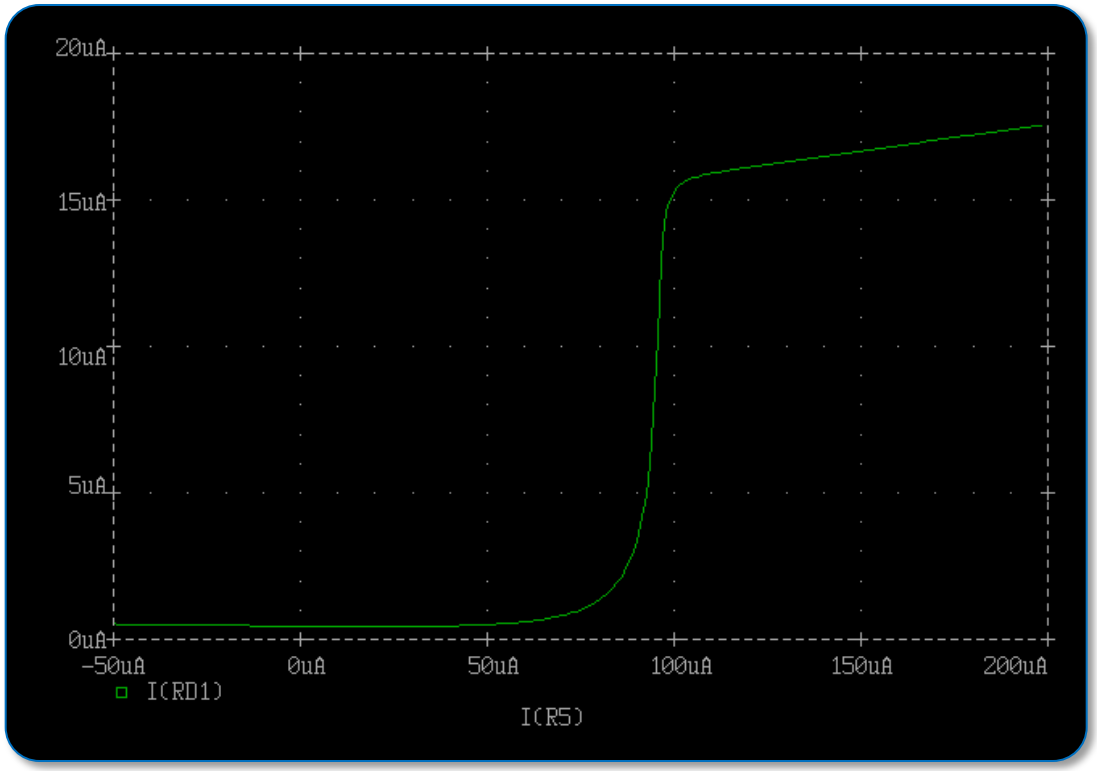
Tools, benchmarks, datasets, models and repository are gradually released to public since 2006!

Teaser: back to 1993 …



Semiconductor neural element - possible base of neural computers and specialized accelerators

Modeling and understanding brain functions



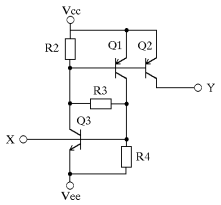
My problem with modeling:

- **Slow**
- **Unreliable**
- **Costly**

Motivation: back to basics

Researhers

Task



Solution

User requirements:

most common:

minimize all costs (time, power consumption, price, size, faults, etc)

guarantee real-time constraints (bandwidth, QOS, etc)

Decision
(depends on user requirements)

Available choices (solutions)

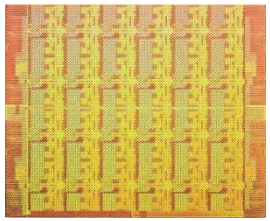
Result

Service/application providers

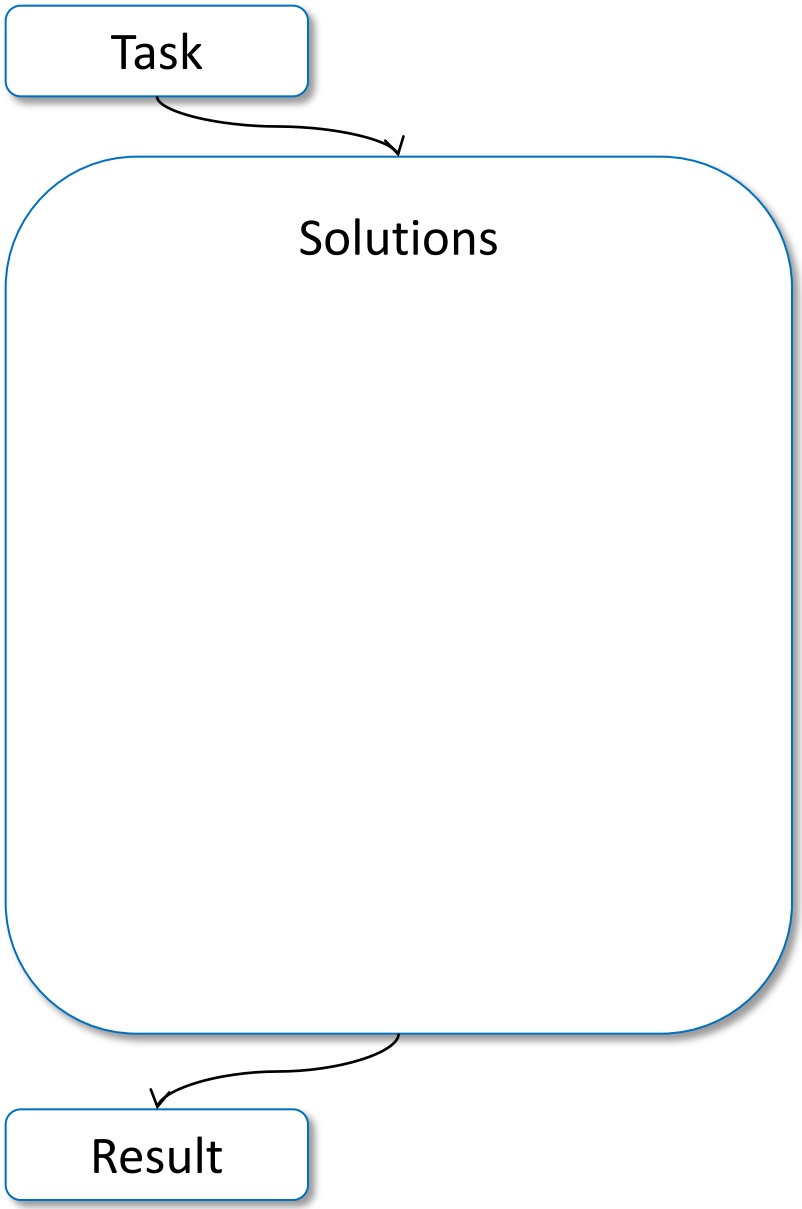
(supercomputing, cloud computing, mobile systems)

Should provide choices and help with decisions

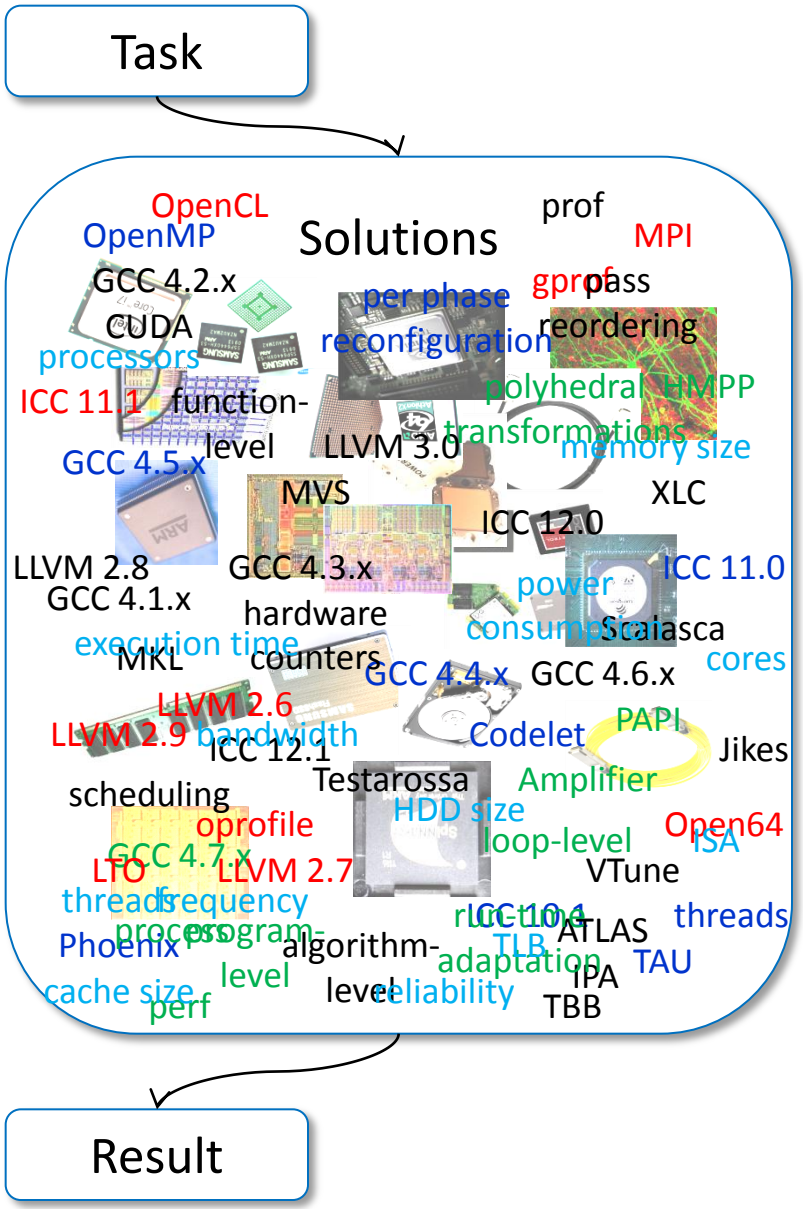
Hardware and software designers



Problems in computer engineering



Problems in computer engineering



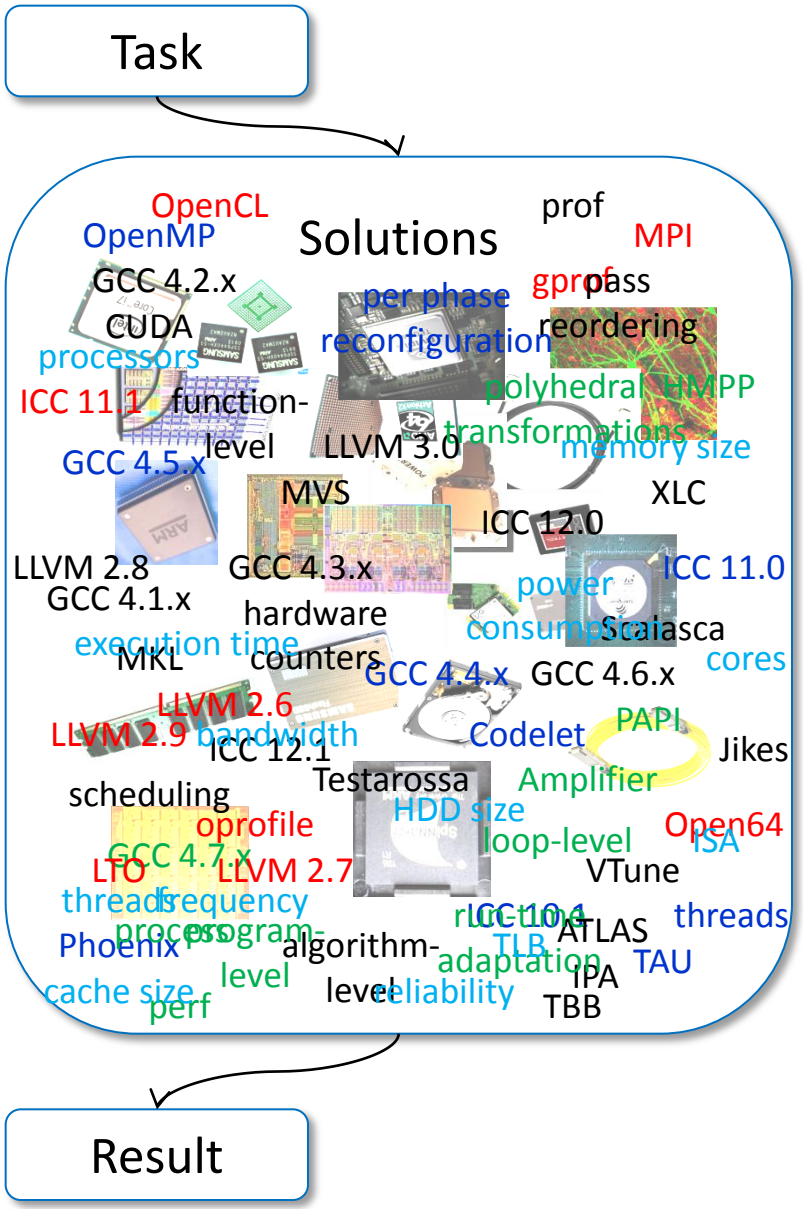
1) Rising complexity of computer systems:
   too many design and optimization choices

2) Performance is not anymore the only requirement:
   multiple user objectives vs choices
   benefit vs optimization time

3) Complex relationship and interactions between ALL software and hardware components (co-design).

4) Too many tools with non-unified interfaces changing from version to version:
   technological chaos

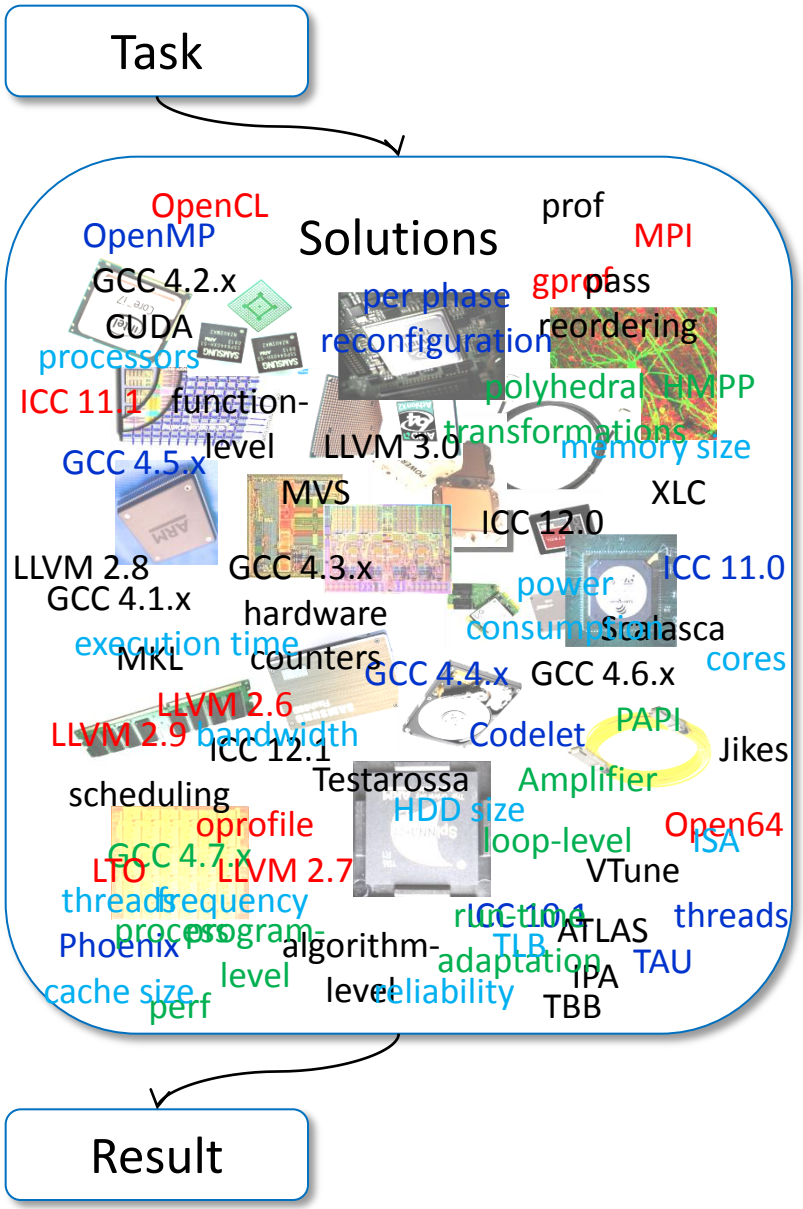
Problems in computer engineering



Challenges for end-users and companies:

- finding the right solution for end-user is extremely challenging
- everyone is lost in choices
- dramatic increase in development time
- low ROI
- underperforming systems
- waste of energy
- ad-hoc, repetitive and error-prone manual tuning
- **slowing innovation in science and technology**

Problems in computer engineering



Challenges for end-users and companies:

- finding the right solution for end-user is extremely challenging
- everyone is lost in choices
- dramatic increase in development time
- low ROI
- underperforming systems
- waste of energy
- ad-hoc, repetitive and error-prone manual tuning
- **slowing innovation in science and technology**

Understanding and modeling of the overall relationship between end-user algorithms, applications, compiler optimizations, hardware designs, data sets and run-time behavior became simply infeasible!

Attempts to solve these problems: auto-tuning

Task

Treat computer system as a black box

Result

Algorithm

Application

Compilers and auxiliary tools

Binary and libraries

Data set

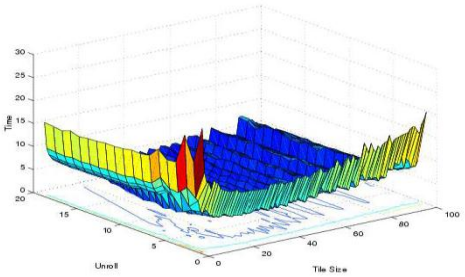
State of the system

Run-time environment

Architecture

Use auto-tuning:

Explore multiple choices empirically: learn behavior of computer systems across executions



Covered all components in the last 2 decades and showed high potential but …

Attempts to solve these problems: auto-tuning

Auto-tuning shows high potential for nearly 2 decades but still far from the mainstream in production environments.

Why?

- Optimization spaces are large and non-linear with many local minima
- Exploration is slow and ad-hoc (random, genetic, some heuristics)
- Only a few benchmarks are considered
- Often the same (one) dataset is used
- Only part of the system is taken into account
(rarely reflect behavior of the whole system)
- No knowledge sharing

Attempts to solve these problems: machine learning

Task

Treat computer system as a black box

Result

Algorithm

Application

Compilers and auxiliary tools

Binary and libraries

Data set

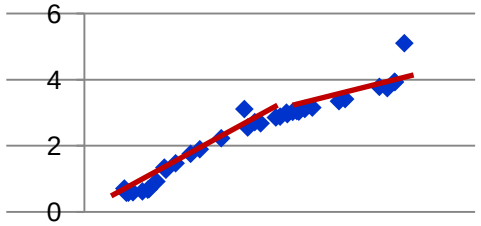
State of the system

Run-time environment

Architecture

Use machine learning to speed up exploration

Apply predictive modeling to suggest profitable solutions based on properties of a task and a system



Covered all components in the last decade and showed high potential but …

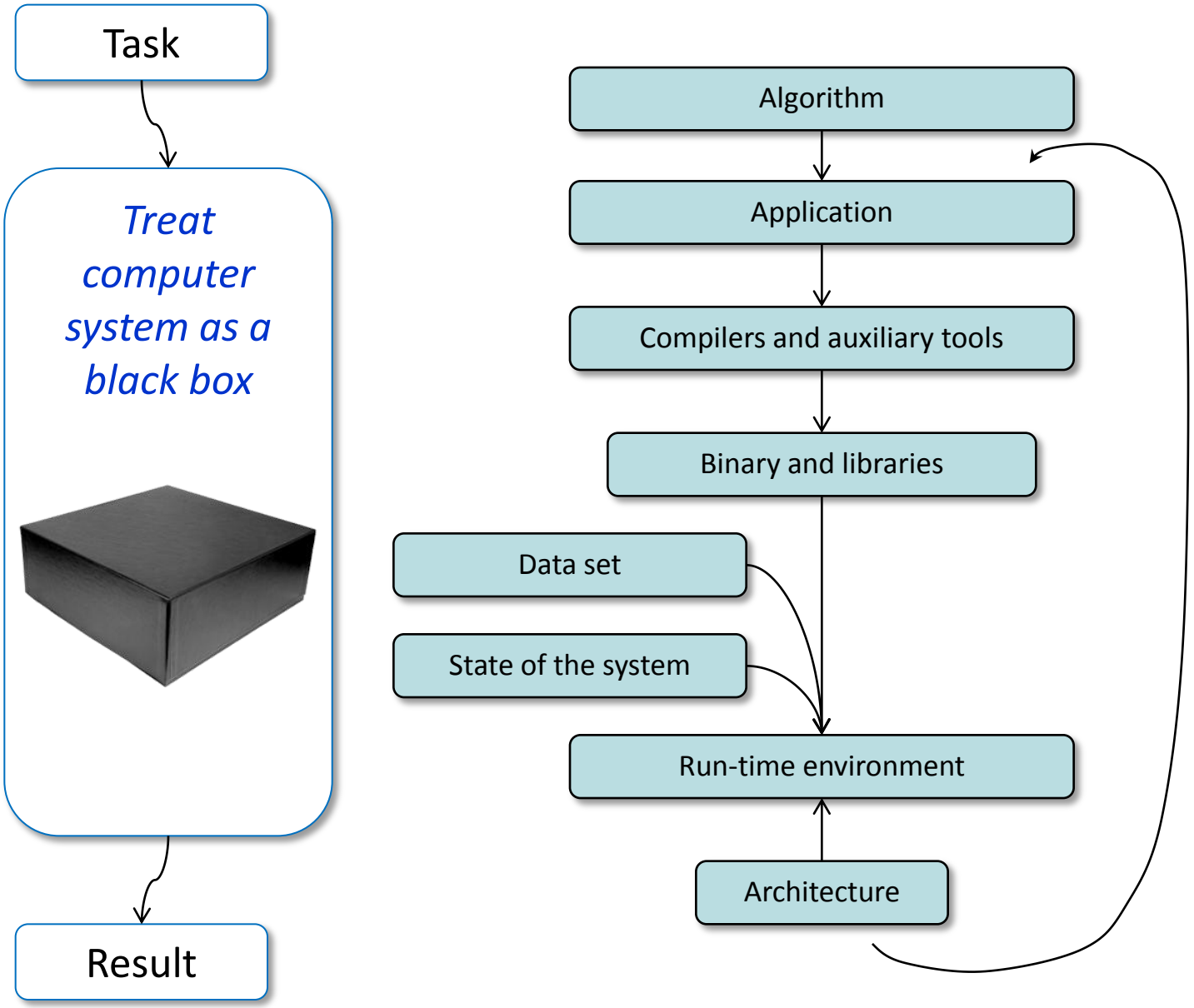
Attempts to solve these problems: machine learning

Machine learning (classification, predictive modeling) shows high potential during past decade but still far from the mainstream.

Why?

- Selection of machine learning models and right properties is non-trivial: ad-hoc in most of the cases
- Limited training sets
- Only part of the system is taken into account (rarely reflect behavior of the whole system)
- No knowledge sharing

Attempts to solve these problems: co-design



Co-design:

Explore choices and behavior of the whole system.

Showed high potential in the last years but …

Attempts to solve these problems: co-design

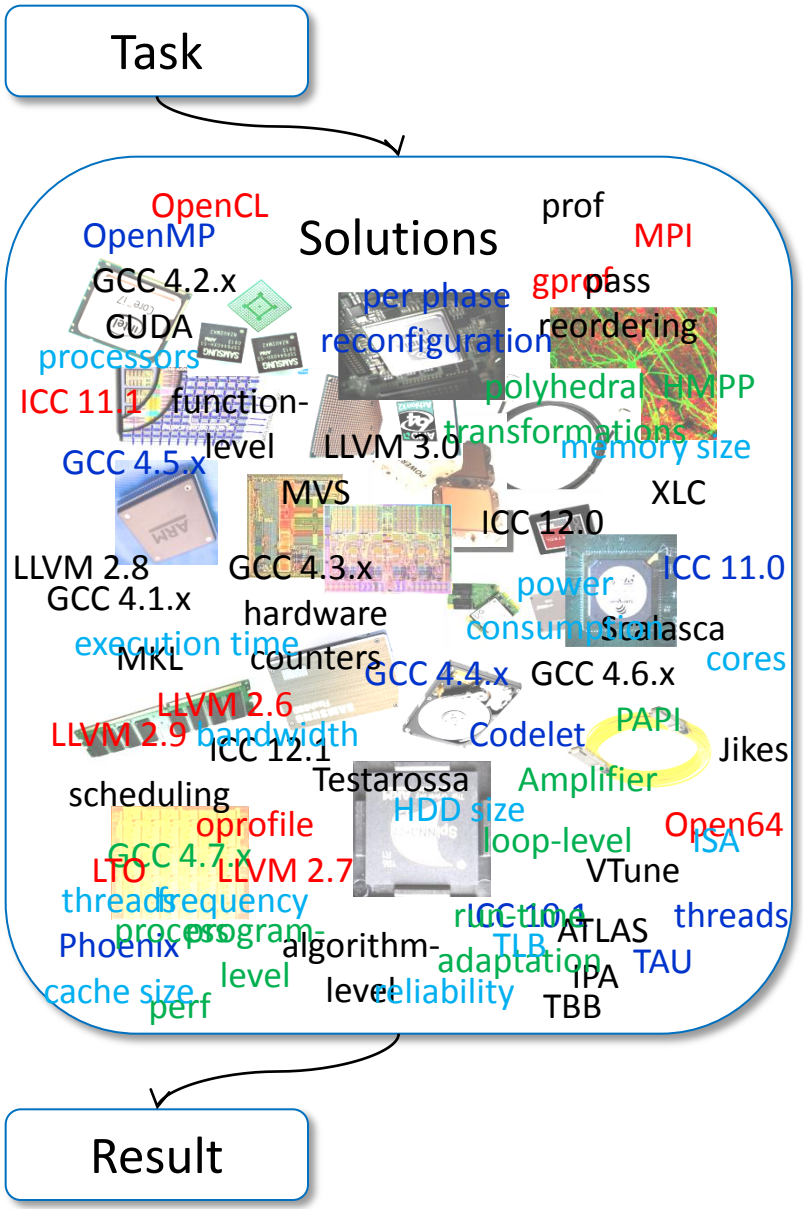
Co-design is currently a buzz word and a hot research topic

but still far from the mainstream.

Why?

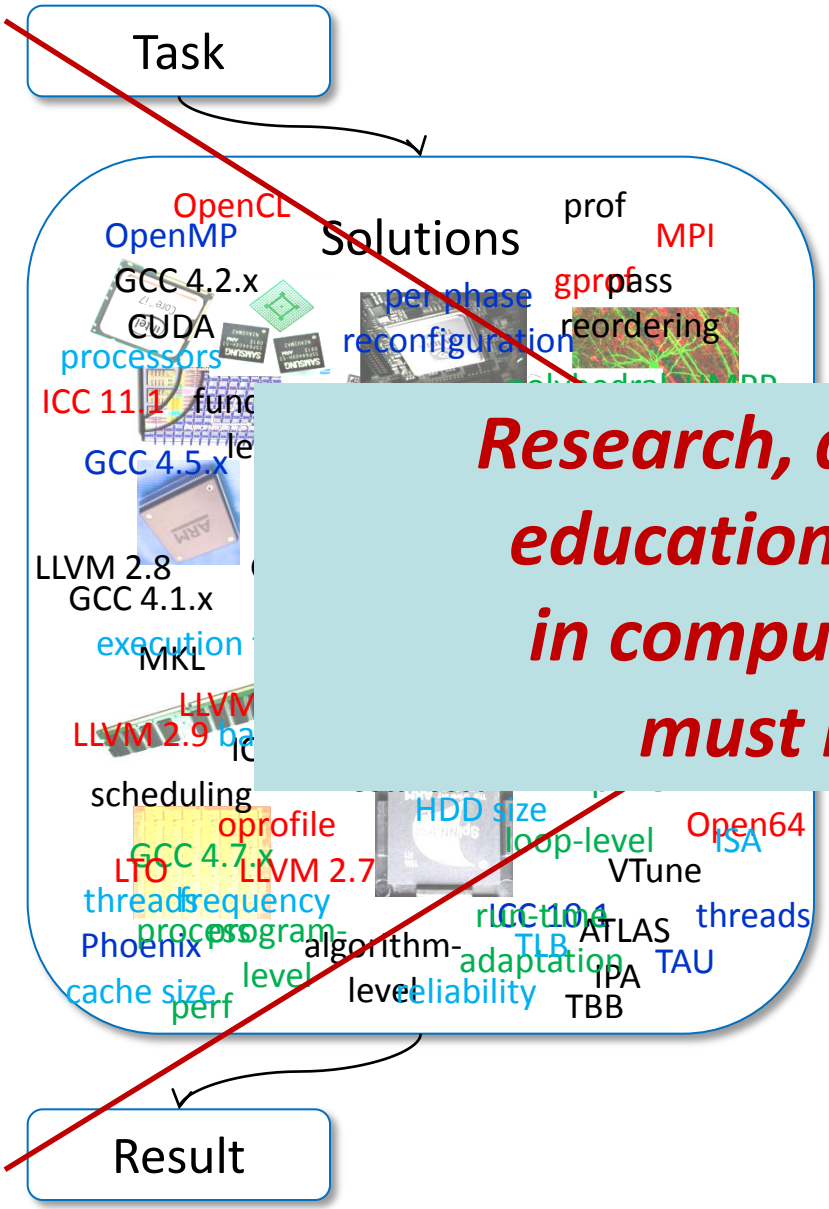
- Even more choices to explore and analyze
- Often impossible to expose tuning choices or obtain characteristics at all levels
- Limited training sets
- Still no knowledge sharing

Problems in academic research



- Too much time wasted on very limited ad-hoc individual experimental setups
- Very small part of a system is usually considered - can be very misleading
- Sharing and reproducibility of results is rarely considered or even practically impossible
- Too many papers on the same topics with non-reproducible results (just for academic promotion)
- Slowing innovation in science and technology
- No trust from industry

Problems in academic research



Research, development and educational methodology in computer engineering must be revisited!

- Too much time wasted on very limited ad-hoc individual experimental setups
- Very small part of a system is usually
- slowing innovation in science and technology
- no trust from industry

Can we crowdsource auto-tuning? My main focus since 2004

Got stuck with a limited number of benchmarks, datasets, architectures and a large number of optimizations and generated data;

could not validate data mining and machine learning techniques

Needed dramatically new approach!

Millions of users run realistic applications on different architectures with different datasets, run-time systems, compilers, optimizations!




Can we leverage their experience and computational resources?

Can we build public repository of knowledge?

Attempts to solve these problems: optimization repositories

Simply too time consuming and costly to build, support and extend particularly with ever changing tools, interfaces, benchmarks, data sets, properties, models, etc.

Usually no public funding for such activities up to now.

Only big companies or projects can afford to build and support their own big repositories but they are either not public (Google, Intel, IBM, ARM) or used as a simple storage of information (SciDAC, SPEC).

Furthermore, public data and tools may cause competition.

Crowdsourcing design and optimization of computer systems

EU MILEPOST project (2006-2009):

We have proposed and started developing collective methodology and infrastructure to crowdsource auto-tuning (cTuning) :

- repository, auto-tuning and machine learning is an integral part of co-design
- repository is dynamically evolving and contains all encountered benchmarks, data sets, tools, codelets, optimized binaries and libraries, choices, properties, characteristics, predictive models, decision trees
- repository and infrastructure are distributed among many users and can automatically exchange information about
  - unexplored choices
  - optimization areas with high variability
  - optimal predictive models
  - abnormal behavior to focus further exploration and validate or improve classification and models

Crowdsourcing design and optimization of computer systems

EU MILEPOST project (2006-2009):

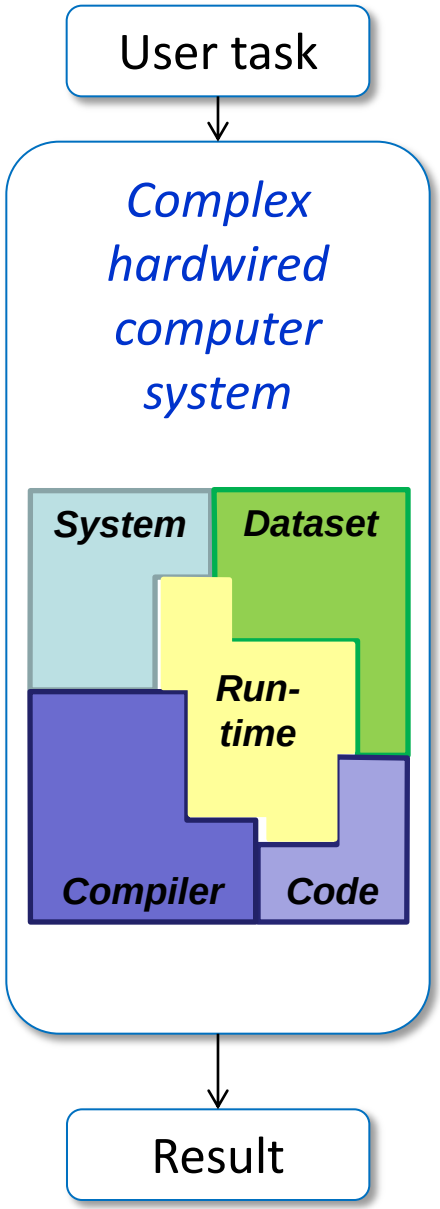
We have proposed and started developing collective methodology and infrastructure to crowdsource auto-tuning (cTuning) :

- repository, auto-tuning and machine learning is an integral part of co-design
- reposito ed benchma d libraries, choices, on trees
- reposito ers and can automat
  - unexp
  - optimization areas with high variability
  - optimal predictive models
  - abnormal behavior to focus further exploration and validate or improve classification and models

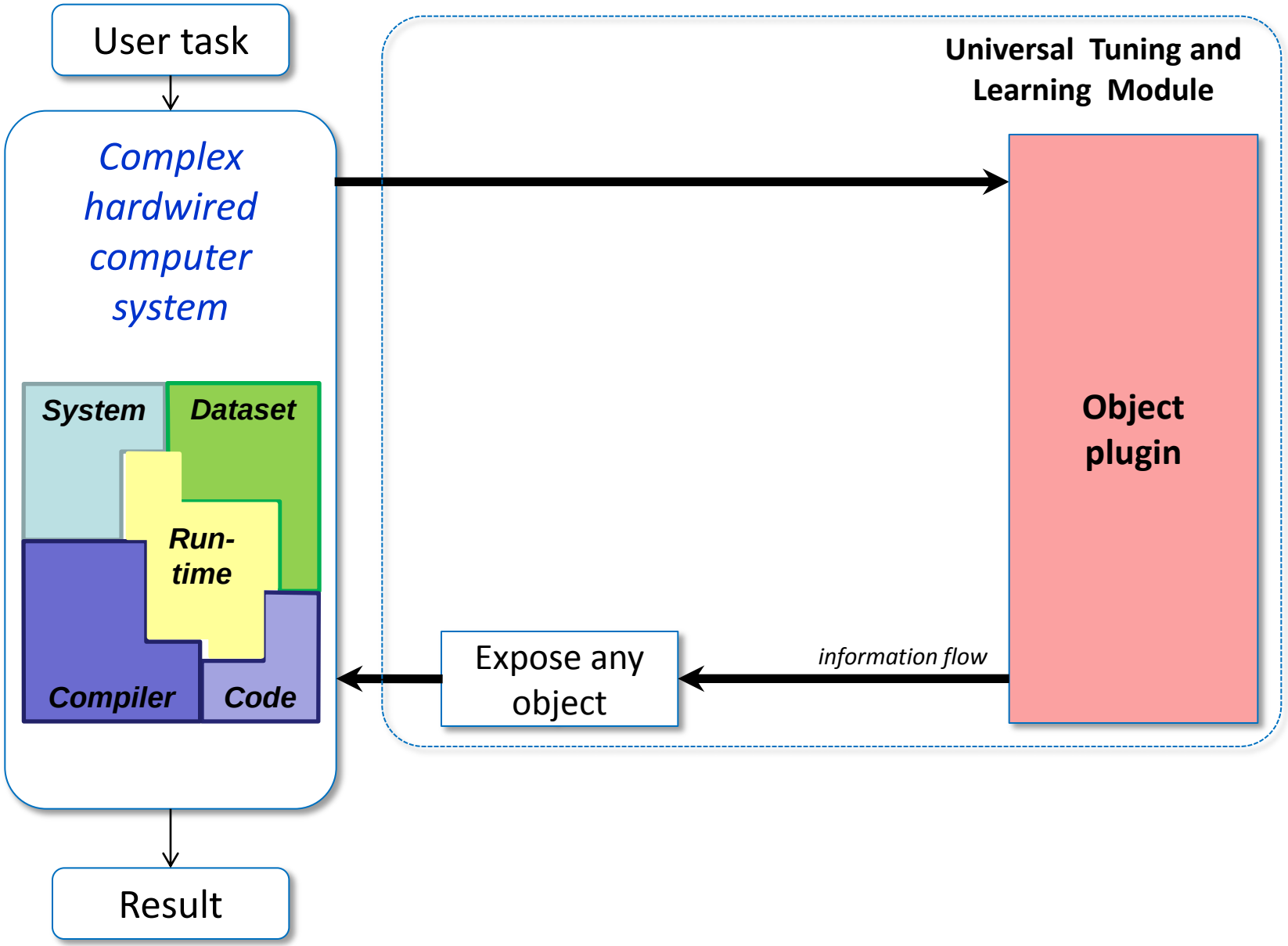
Main challenge:

how to make it simple, extensible and implement with very limited funding and 1..2 researchers instead of redesigning the whole software/hardware stack (like in IBM's Liquid Metal project)?

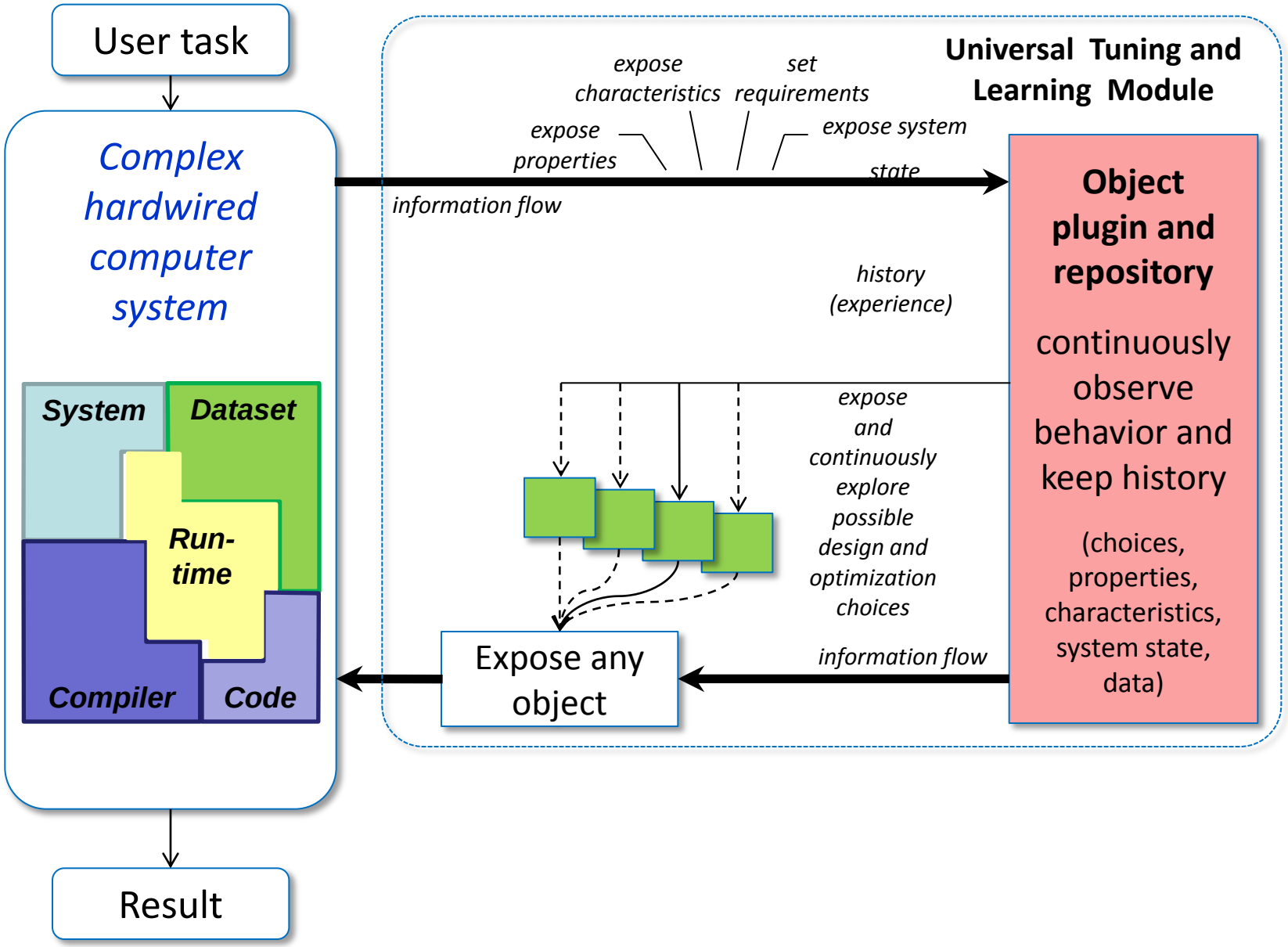
Gradual decomposition, parameterization, tuning and learning of computer systems



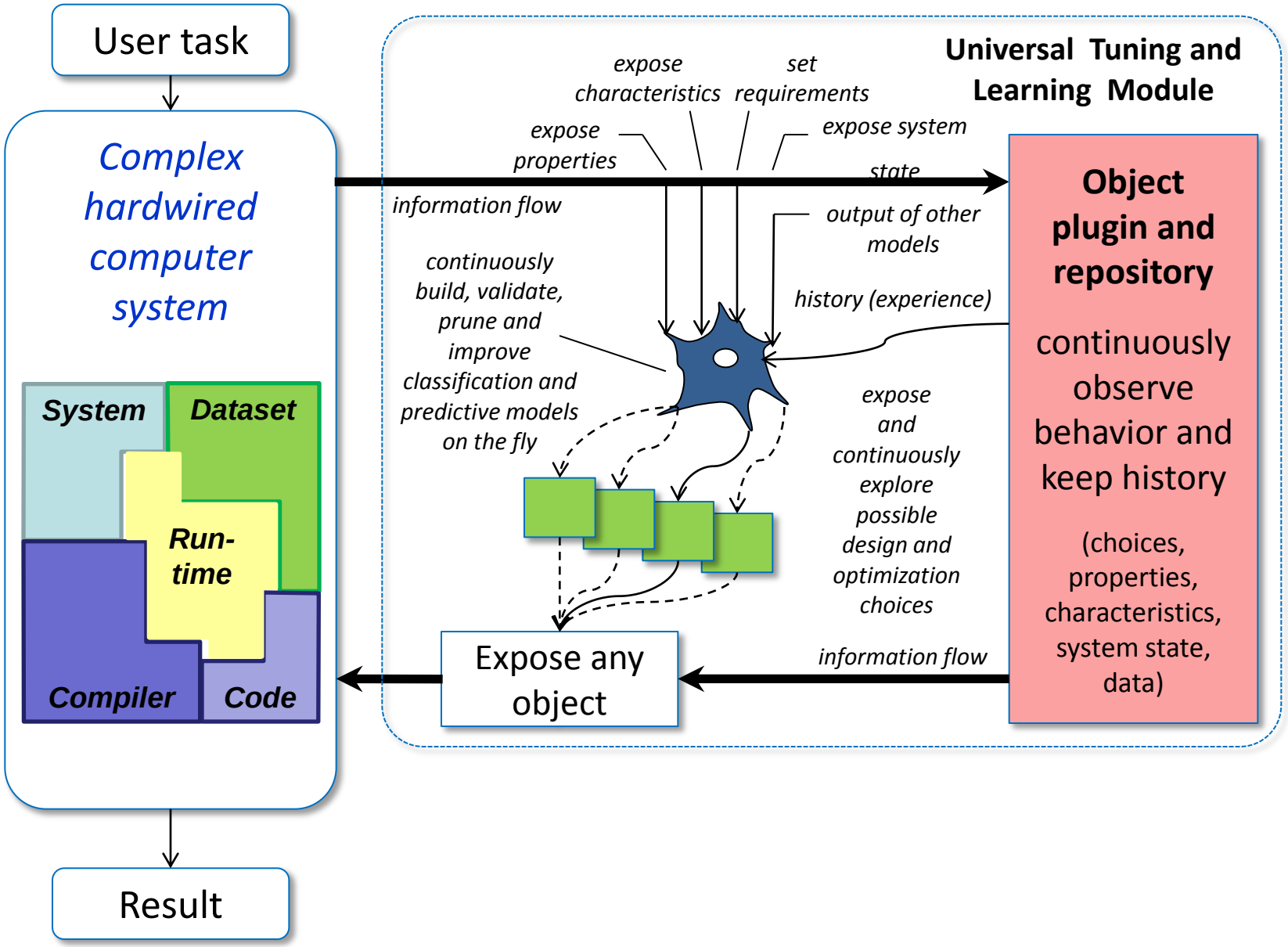
Gradual decomposition, parameterization, tuning and learning of computer systems



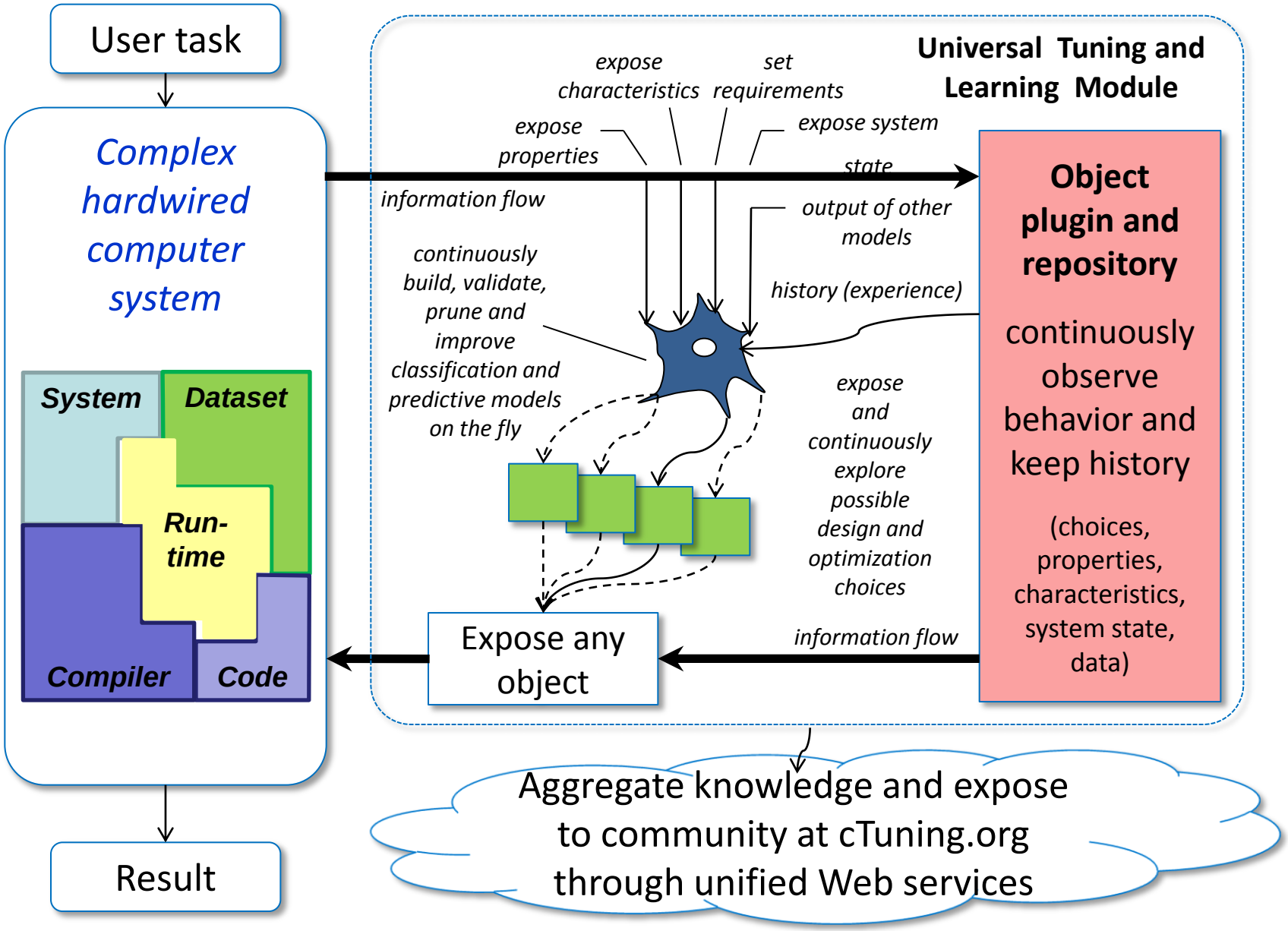
Gradual decomposition, parameterization, tuning and learning of computer systems



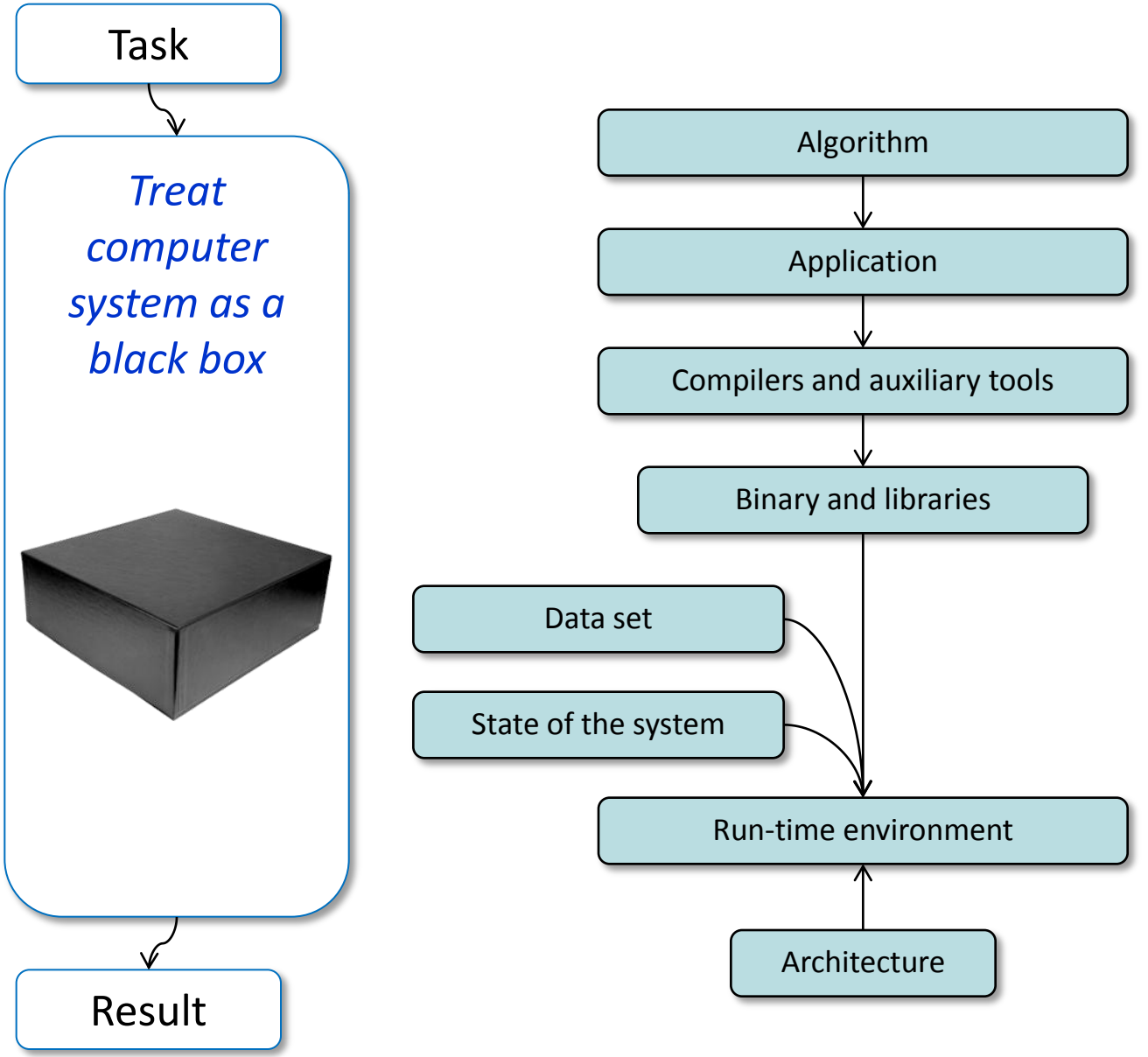
Gradual decomposition, parameterization, tuning and learning of computer systems



Gradual decomposition, parameterization, tuning and learning of computer systems



Top-down decomposition of computer system to keep complexity under control



Top-down decomposition of computer system to keep complexity under control

Task

Treat computer system as a black box

Result

cTuning$_3$ framework aka Collective Mind

- Algorithm — Repo/models
- Application — Repo/models
- Compilers and auxiliary tools — Repo/models
- Binary and libraries — Repo/models
- Data set — Repo/models
- State of the system — Repo/models
- Run-time environment — Repo/models
- Architecture — Repo/models

Top-down decomposition of computer system to keep complexity under control

Task

Treat computer system as a black box

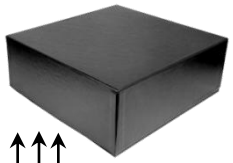
Result

cTuning$_3$ framework aka Collective Mind

- Algorithm — Repo/models
- Application — Repo/models
- Compilers and auxiliary tools — Repo/models
- Binary and libraries — Repo/models
- Data set — Repo/models
- State of the system — Repo/models
- Run-time environment — Repo/models
- Architecture — Repo/models

Light-weight interface to connect modules, data and models

Top-down decomposition of computer system to keep complexity under control



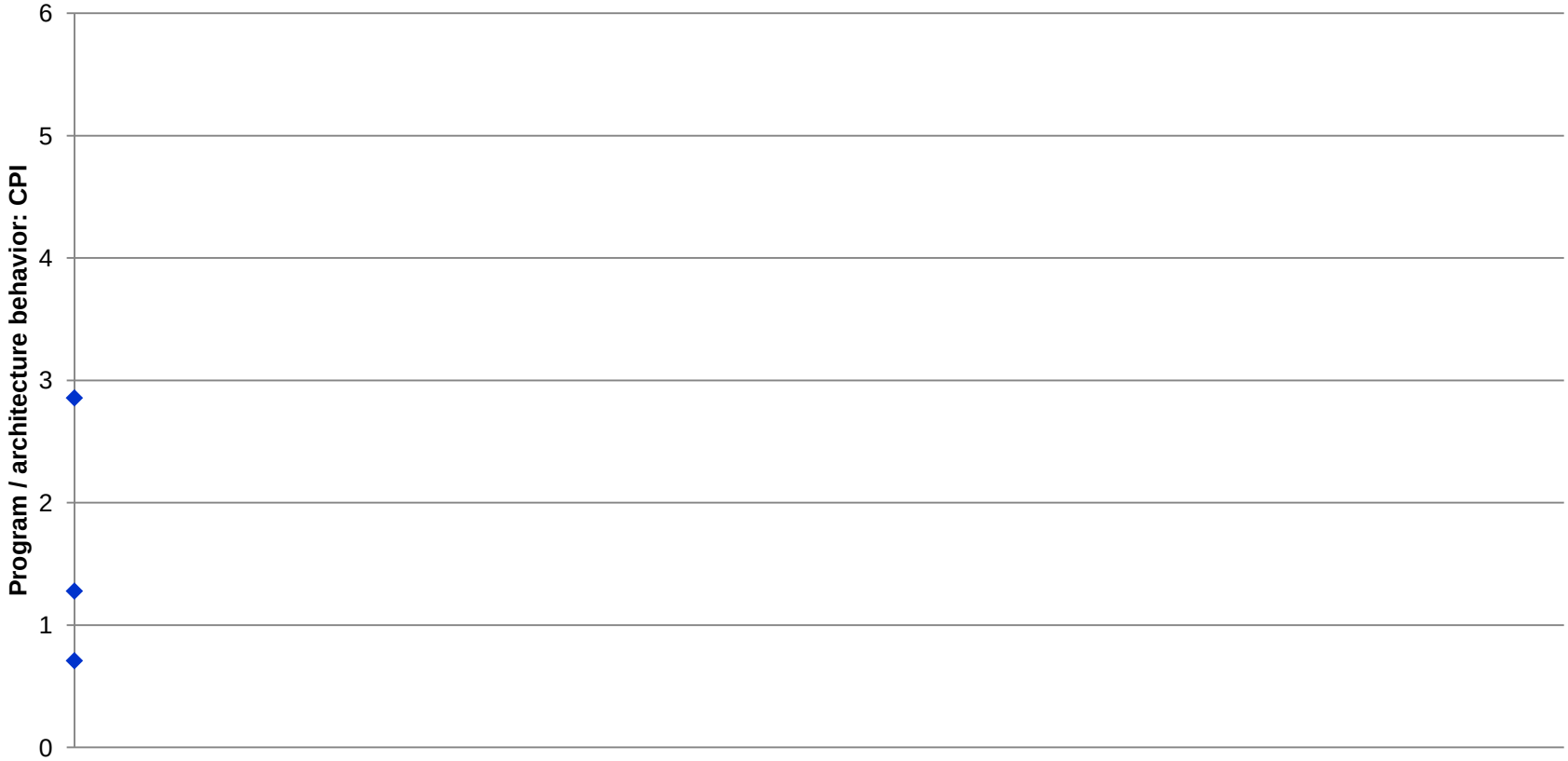
		Gradually expose some characteristics	Gradually expose some properties/choices
Compile Program		time …	compiler flags; pragmas …
Run code	Run-time environment	time; CPI, power consumption …	pinning/scheduling …
	System	cost;	architecture; frequency; cache size…
	Data set	size; values; description …	precision …
Analyze profile		time; size …	instrumentation; profiling …

Start coarse-grain decomposition of a system (detect coarse-grain effects first). Add universal learning modules.

Example of characterizing/explaining behavior of computer systems

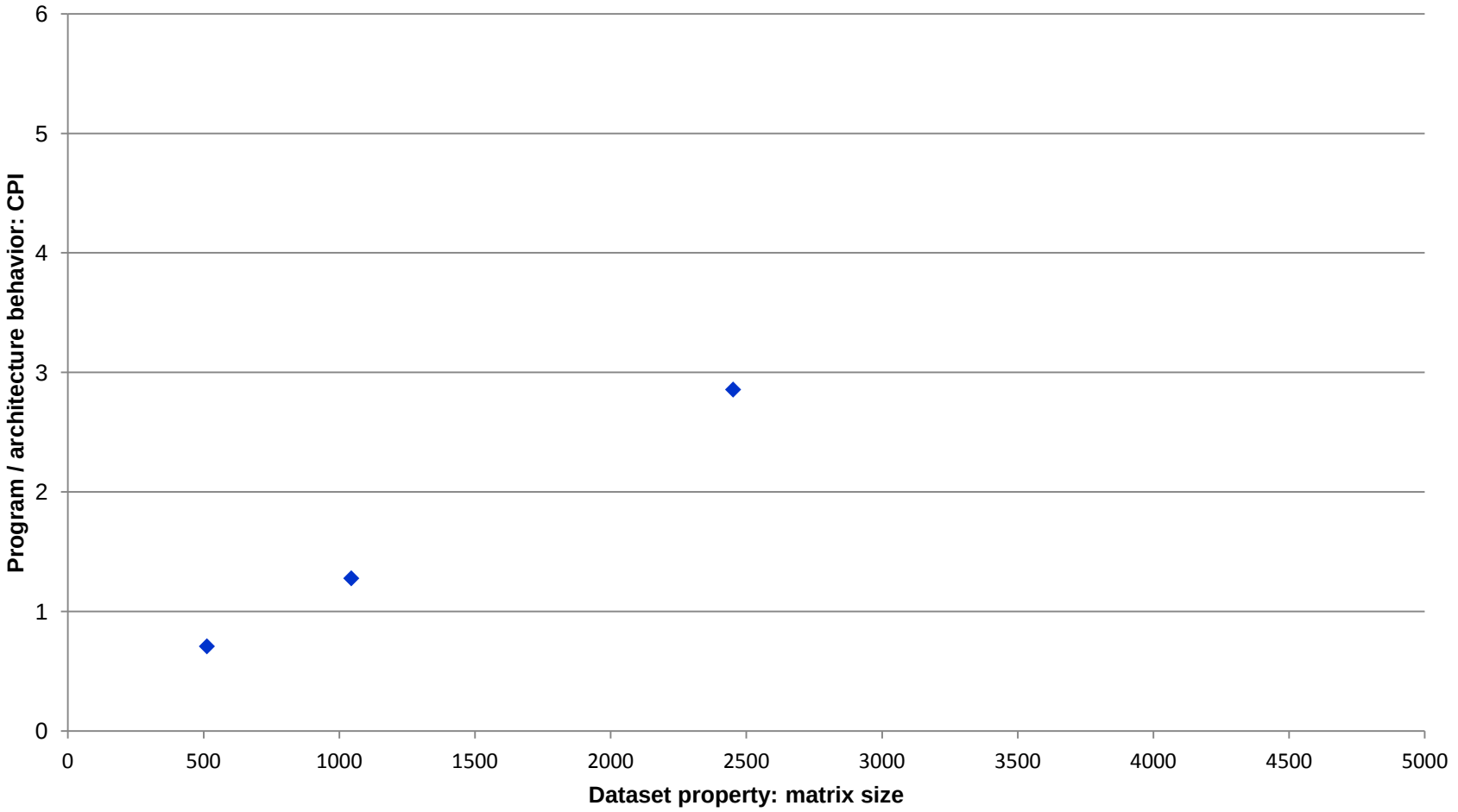
How we can explain the following observations for some piece of code (“codelet object”)?

(LU-decomposition codelet, Intel Nehalem)



Example of characterizing/explaining behavior of computer systems

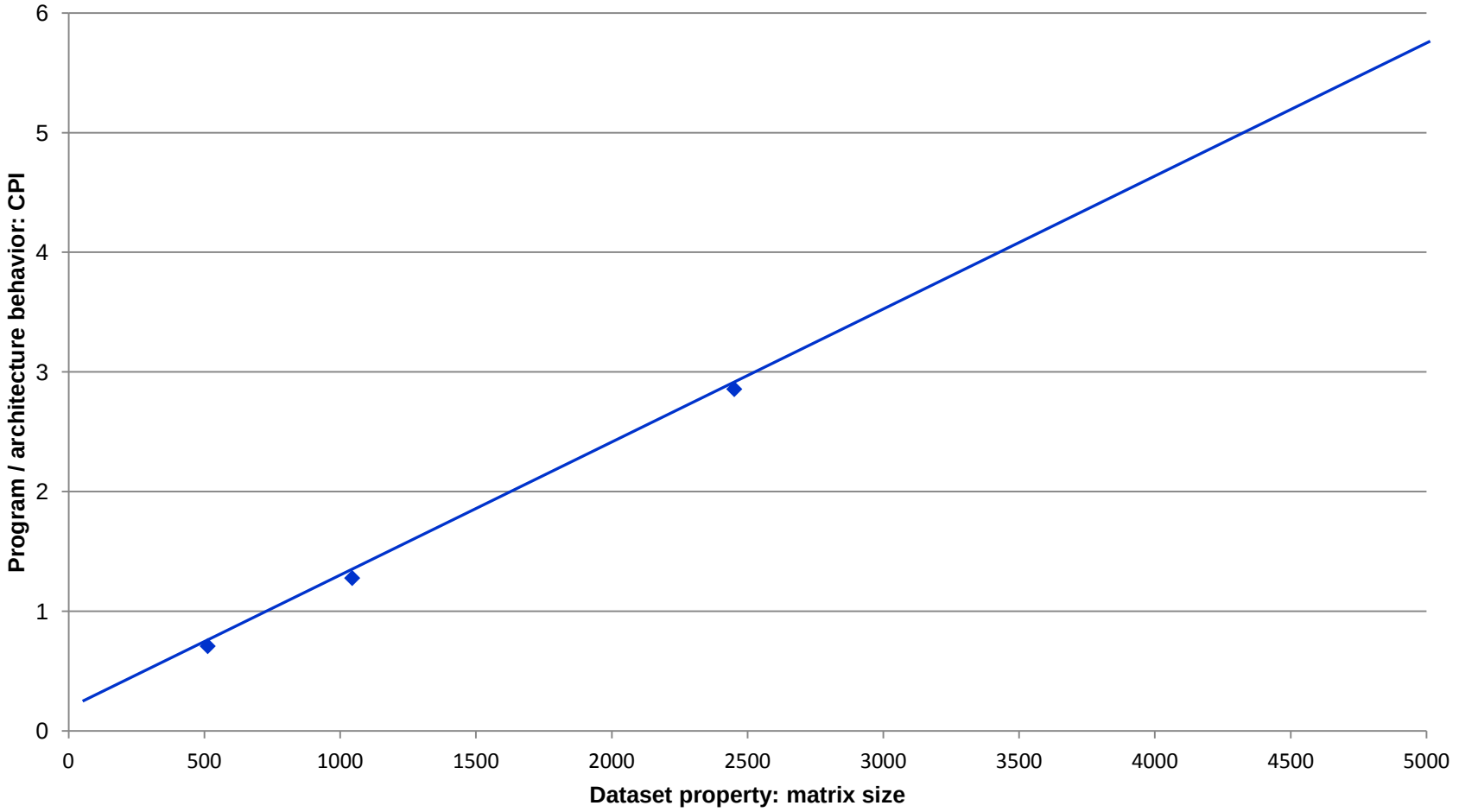
Add 1 property: matrix size



Example of characterizing/explaining behavior of computer systems

Try to build a model to correlate objectives (CPI) and features (matrix size).

Start from simple models: linear regression (detect coarse grain effects)

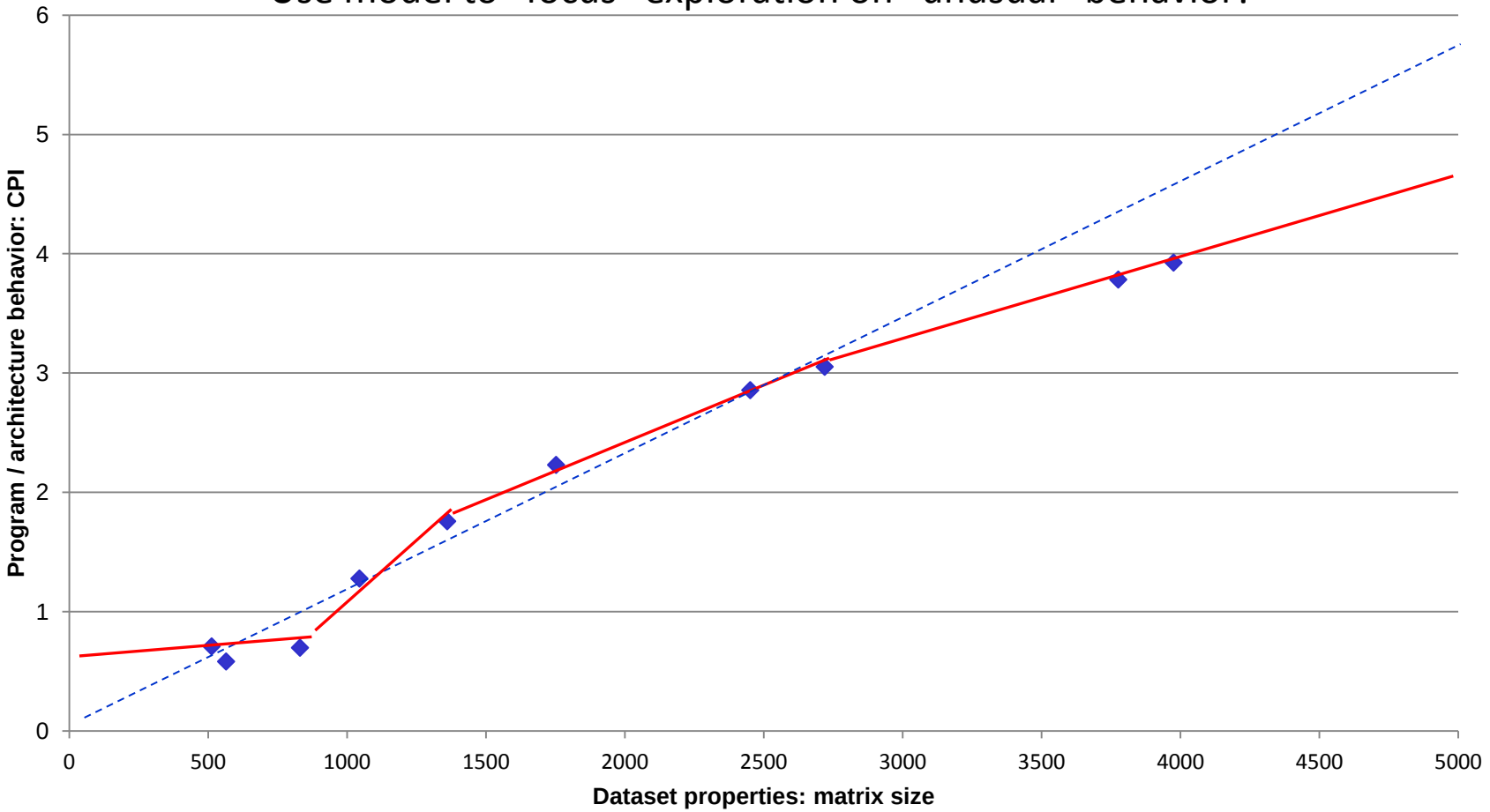


Example of characterizing/explaining behavior of computer systems

If more observations, validate model and detect discrepancies!

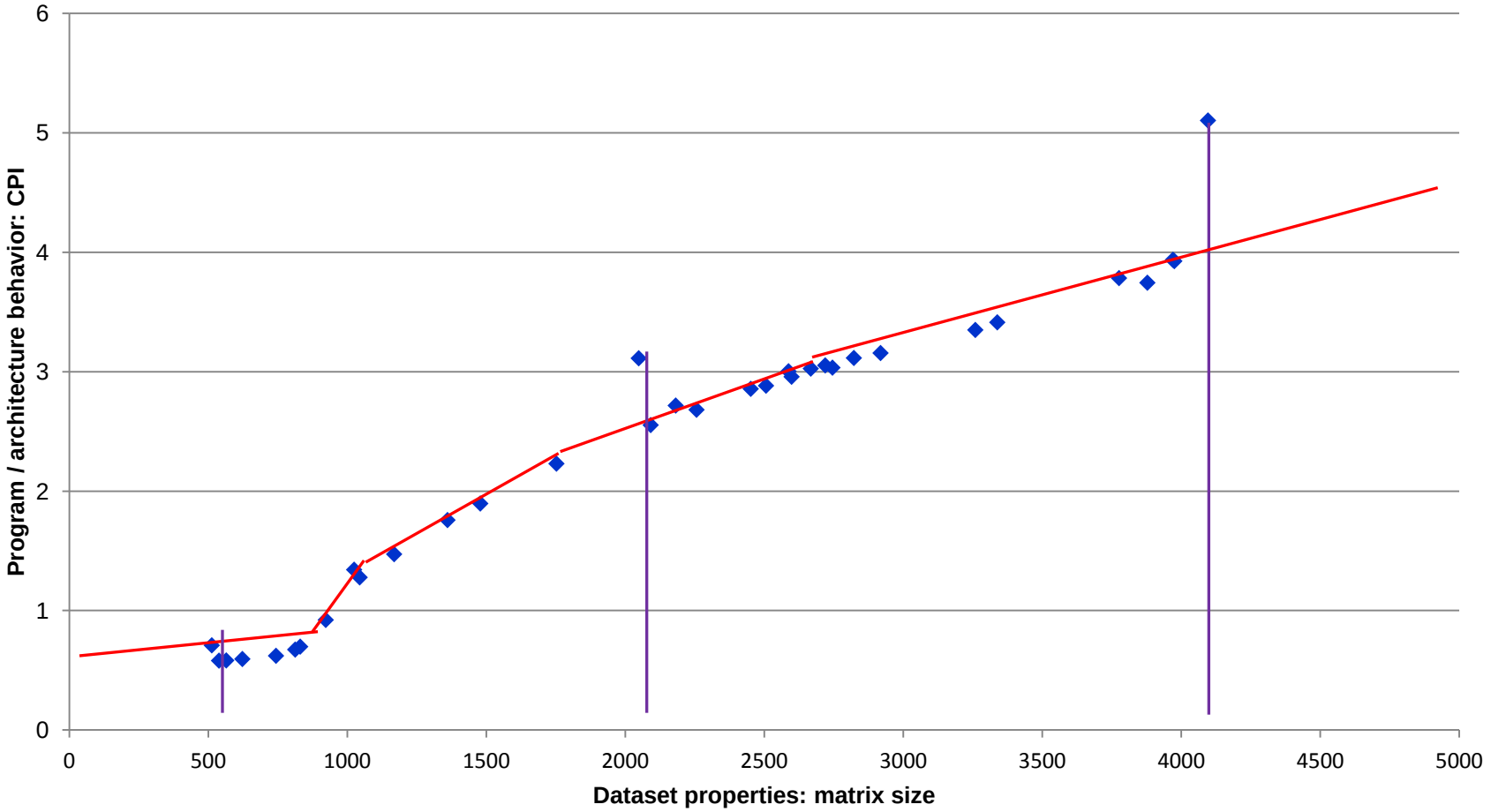
Continuously retrain models to fit new data!

Use model to "focus" exploration on "unusual" behavior!



Example of characterizing/explaining behavior of computer systems

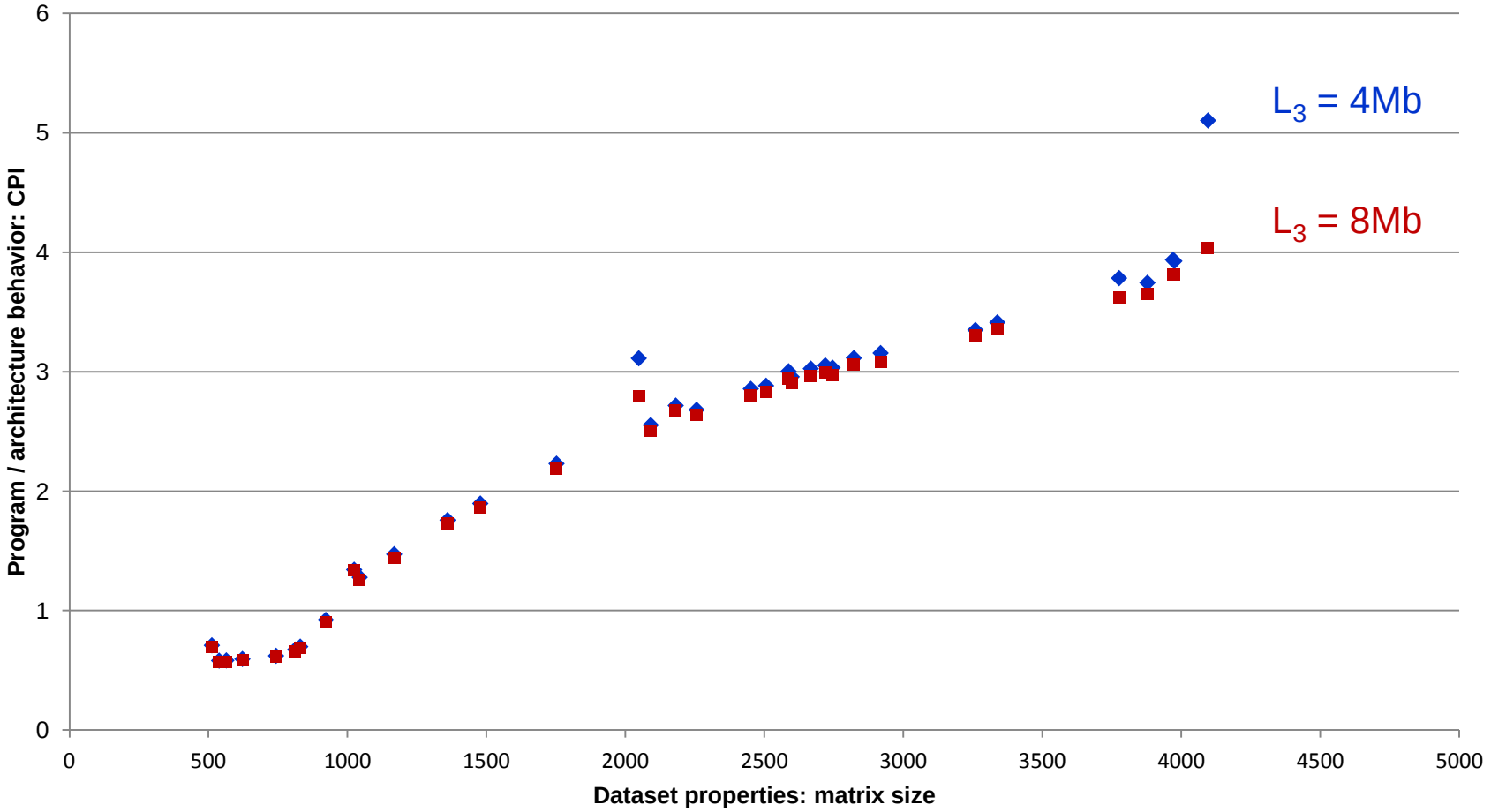
Gradually increase model complexity if needed (hierarchical modeling).
For example, detect fine-grain effects (singularities) and characterize them.



Example of characterizing/explaining behavior of computer systems

Start adding more properties (one more architecture with twice bigger cache)!

Use automatic approach to correlate all objectives and features.



Example of characterizing/explaining behavior of computer systems

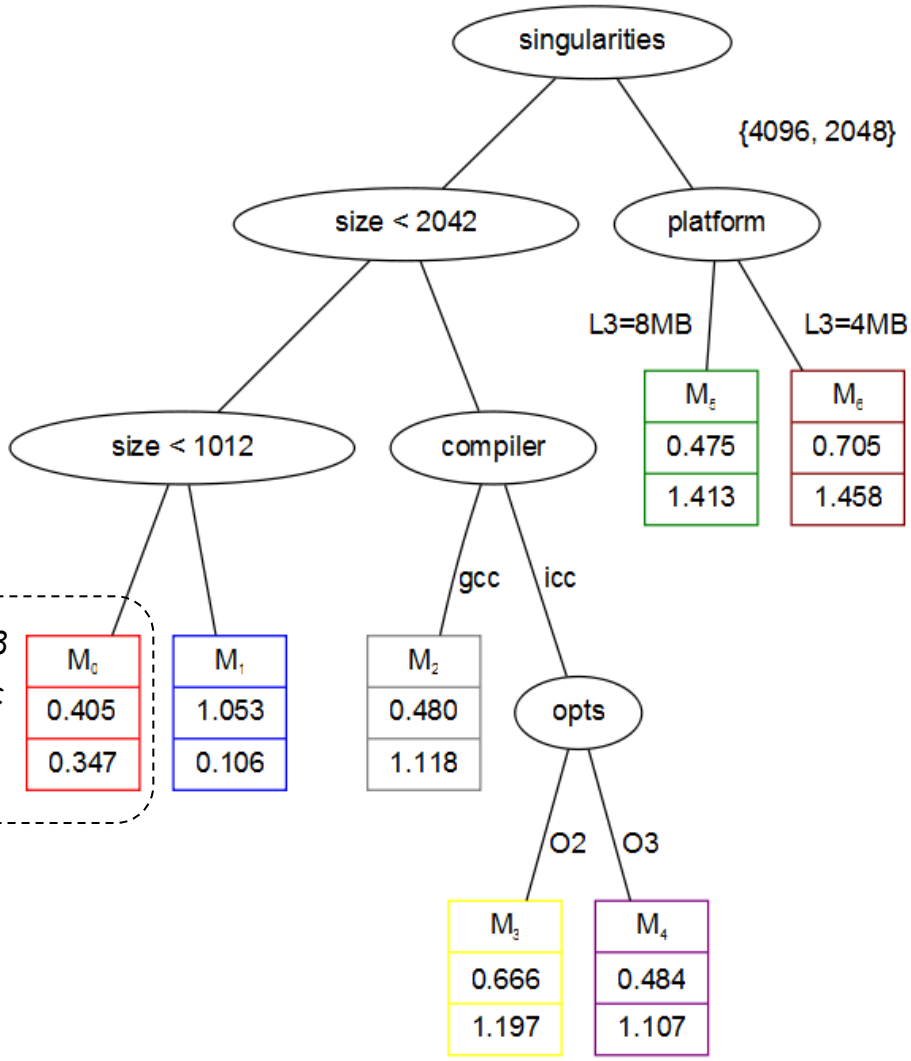
Continuously build and refine classification (decision trees for example) and predictive models on all collected data to improve predictions.

Continue exploring design and optimization spaces (evaluate different architectures, optimizations, compilers, etc.)

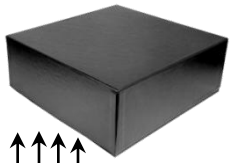
Focus exploration on unexplored areas, areas with high variability or with high mispredict rate of models

cM predictive model module

CPI = ε + 1000 × β × data size



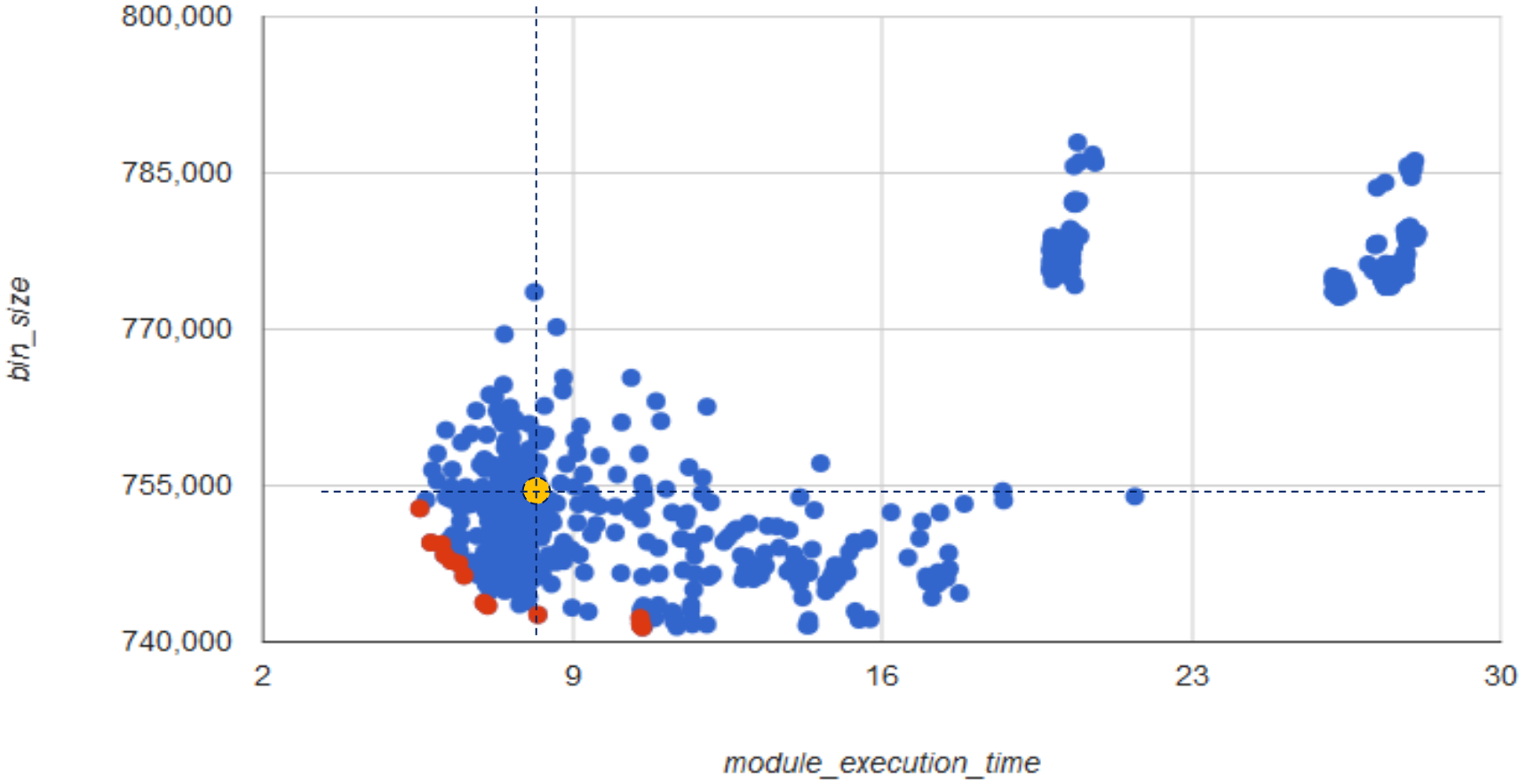
Gradually increasing complexity



				Gradually expose some characteristics	Gradually expose some choices
	Algorithm selection			(time) productivity, variable-accuracy, complexity …	Language, MPI, OpenMP, TBB, MapReduce …
Compile Program				time …	compiler flags; pragmas …
	Code analysis & Transformations			time; memory usage; code size …	transformation ordering; polyhedral transformations; transformation parameters; instruction ordering …
		Process			
		Thread			
		Function			
		Codelet			
		Loop			
			Instruction		
Run code	Run-time environment			time; power consumption …	pinning/scheduling …
	System			cost; size …	CPU/GPU; frequency; memory hierarchy …
	Data set			size; values; description …	precision …
	Run-time analysis			time; precision …	hardware counters; power meters …
	Run-time state			processor state; cache state …	helper threads; hardware counters …
Analyze profile				time; size …	instrumentation; profiling …

Coarse-grain vs. fine-grain effects: depends on user requirements and expected ROI

Multi-objective crowd-tuning using mobile phones



Program:	*cBench: susan corners*	Processor:	*ARM v6, 830MHz*
Compiler:	*Sourcery GCC for ARM v4.6.1*	OS:	*Android OS v2.3.5*
System:	*Samsung Galaxy Y*	Data set:	*MiDataSet #1, image, 600x450x8b PGM, 263KB*

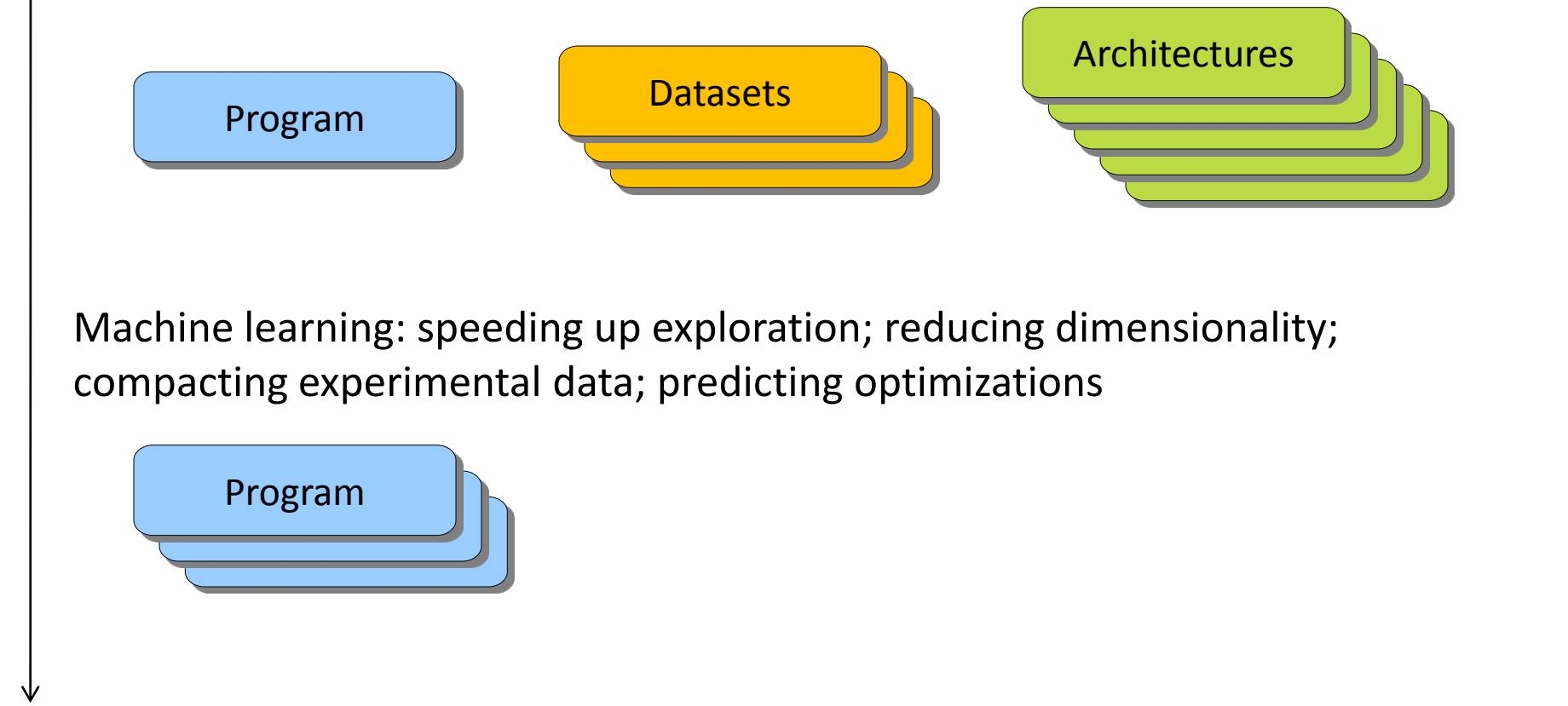
Pool of best optimization classes across programs

-O1 -falign-loops=10 -fpeephole2 -fschedule-insns -fschedule-insns2 -fno-tree-ccp -fno-tree-dominator-opts -funroll-loops
-O1 -fpeephole2 -fno-rename-registers -ftracer -fno-tree-dominator-opts -fno-tree-loop-optimize -funroll-all-loops
-O2 -finline-functions -fno-tree-dce -fno-tree-loop-im -funroll-all-loops
-O2 -fno-guess-branch-probability -fprefetch-loop-arrays -finline-functions -fno-tree-ter
-O2 -fno-tree-lrs
-O2 -fpeephole -fno-peephole2 -fno-regmove -fno-unswitch-loops
-O3 -finline-limit=1481 -falign-functions=64 -fno-crossjumping -fno-ivopts -fno-tree-dominator-opts -funroll-loops
-O3 -finline-limit=64
-O3 -fno-tree-dominator-opts -funroll-loops
-O3 -frename-registers
-O3 -fsched-stalled-insns=19 -fschedule-insns -funroll-all-loops
-O3 -fschedule-insns -fno-tree-loop-optimize -fno-tree-lrs -fno-tree-ter -funroll-loops
-O3 -funroll-all-loops
-O3 -funroll-loops

Statistical filtering of hundreds combinations of flags

Optimization knowledge reuse across programs

Auto-tuning:
systematizing knowledge per program across datasets and architectures



Machine learning: speeding up exploration; reducing dimensionality; compacting experimental data; predicting optimizations

Validate and improve existing predictive modeling techniques

Collecting data from multiple users in a unified way allows to apply various data mining (machine learning) techniques to detect relationship between the behaviour and features of all components of the computer systems

1) Gradually add/expose various program (features). Automate process or use expert knowledge:

MILEPOST GCC with Interactive Compilation Interface:

ft1 - Number of basic blocks in the method

…

ft19 - Number of direct calls in the method
ft20 - Number of conditional branches in the method
ft21 - Number of assignment instructions in the method
ft22 - Number of binary integer operations in the method
ft23 - Number of binary floating point operations in the method
ft24 - Number of instructions in the method
…
ft54 - Number of local variables that are pointers in the method
ft55 - Number of static/extern variables that are pointers in the method

Code patterns:

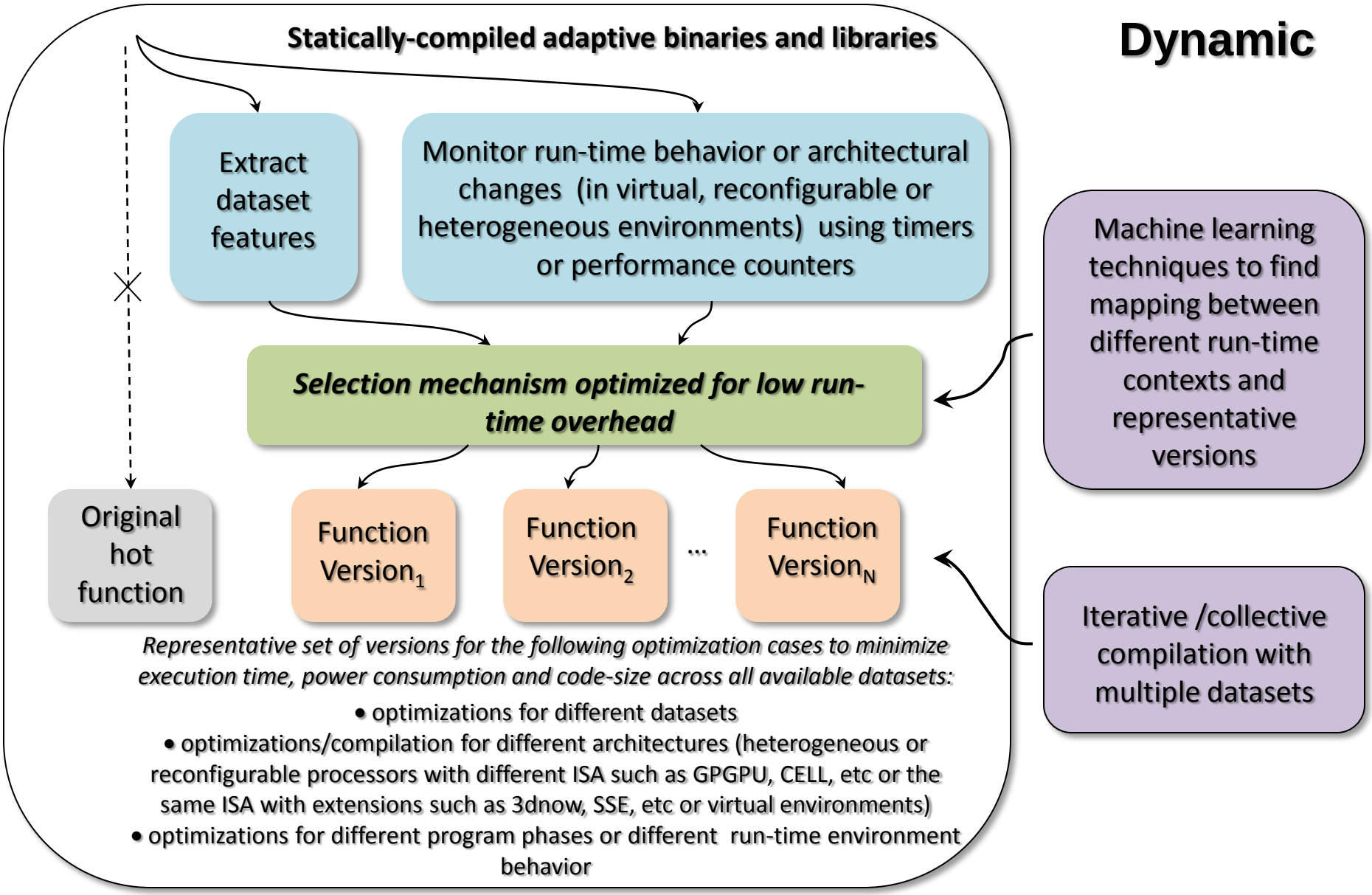
```
for                    F
  for                  F
    for                F
                …
      load …           L
      mult …           A
      store …          S
                …
```

2) Collect run-time, architecture and OS properties (currently hardware counters and architecture descriptions)

3) Correlate features and objectives in cTuning using nearest neighbor classifiers, decision trees, SVM, fuzzy pattern matching, etc.

4) Given new program, dataset, architecture, predict behavior based on prior knowledge!

Statically enable dynamic optimizations (Split-compilation)



cTuning enabled the following research works

• Grigori Fursin, et al. **A Practical Method For Quickly Evaluating Program Optimizations.** Proceedings of the 1st International Conference on High Performance Embedded Architectures & Compilers (HiPEAC 2005), number 3793 in LNCS, pages 29-46, Barcelona, Spain, November 2005

• *Grigori Fursin.* ***Collective Tuning Initiative: automating and accelerating development and optimization of computing systems.*** *Proceedings of the GCC Summit'09, Montreal, Canada, June 2009*

• Victor Jimenez, Isaac Gelado, Lluis Vilanova, Marisa Gil, Grigori Fursin and Nacho Navarro. **Predictive runtime code scheduling for heterogeneous architectures.** Proceedings of the International Conference on High Performance Embedded Architectures & Compilers (HiPEAC 2009), Paphos, Cyprus, January 2009

• Lianjie Luo, Yang Chen, Chengyong Wu, Shun Long and Grigori Fursin. **Finding representative sets of optimizations for adaptive multiversioning applications.** 3rd International Workshop on Statistical and Machine Learning Approaches Applied to Architectures and Compilation (SMART'09) co-located with HiPEAC'09, Paphos, Cyprus, January 2009

• Grigori Fursin and Olivier Temam. **Collective Optimization: A Practical Collaborative Approach.** ACM Transactions on Architecture and Code Optimization (TACO), December 2010, Volume 7, Number 4, pages 20-49

• Grigori Fursin et al. **MILEPOST GCC: machine learning enabled self-tuning compiler.** International Journal of Parallel Programming (IJPP), June 2011, Volume 39, Issue 3, pages 296-327

• Yang Chen, Shuangde Fang, Yuanjie Huang, Lieven Eeckhout, Grigori Fursin, Olivier Temam, Chengyong Wu: **Deconstructing iterative optimization.** ACM TACO 9(3): 21 (2012)

New publication model: reproducibility and validation by the community

• Grigori Fursin, et al. **A Practical Method For Quickly Evaluating Program Optimizations.** Proceedings of the 1st International Conference on High Performance Embedded Architectures & Compilers (HiPEAC 2005), numb

• *Grigori*
optimiza *2009*

• Victor J **ictive**
runtime nference
on High iary
2009

• Lianjie **ets of**
optimiza cal and
Machine with
HiPEAC'0

• Grigori . ACM
Transact r 4,
pages 20

• Grigori F onal
Journal of Pa

Share

Validate

Improve

Explore

Model

Have fun!

Most of the tools, benchmarks, datasets, models are now available online at cTuning.org or added to mainline GCC or available in some commercial tools

• Yang Chen, Shuangde Fang, Yuanjie Huang, Lieven Eeckhout, Grigori Fursin, Olivier Temam, Chengyong Wu: **Deconstructing iterative optimization.** ACM TACO 9(3): 21 (2012)

Collective Mind: new collaborative R&D initiative and publication model

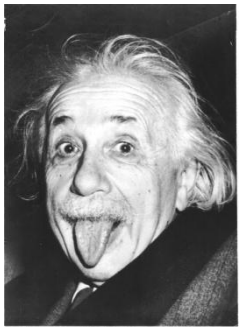
Address rising complexity of the design and optimization of computer systems through practical collaborative analysis, auto-tuning, machine learning and crowdsourcing!

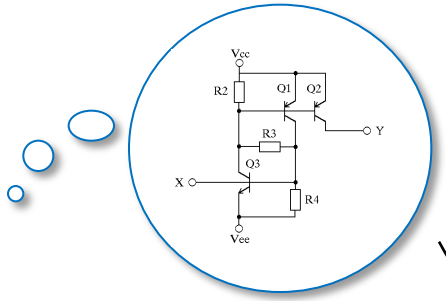
- *building collaborative extensible repository and infrastructure to collect statistics, benchmarks, codelets, tools, data sets and predictive models from the community*
- *preparing new publication model (workshops, conferences, journals) with validation of experimental results by the community*
- *systematizing and unifying optimization, design space exploration and run-time adaptation techniques (co-design and auto-tuning)*
- *evaluating various data mining, classification and predictive modeling techniques for off-line and on-line auto-tuning*

http://cTuning.org/making-computer-engineering-a-science-2013

Collective Mind: my wish since 1999

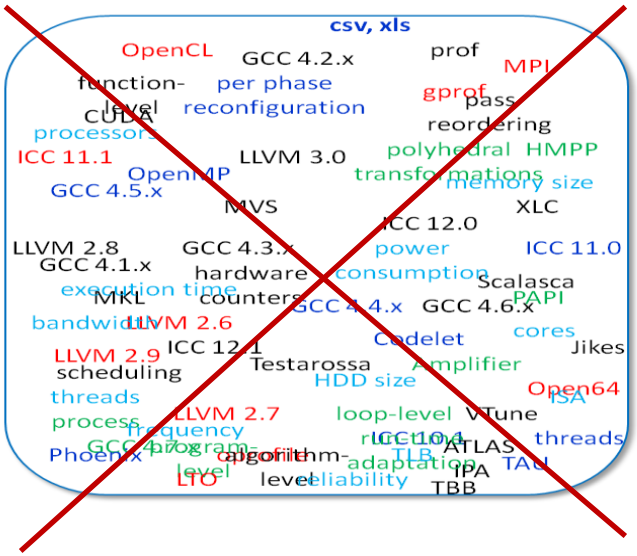
New idea?





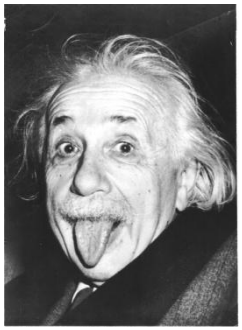
Current non-systematic, tedious R&D:

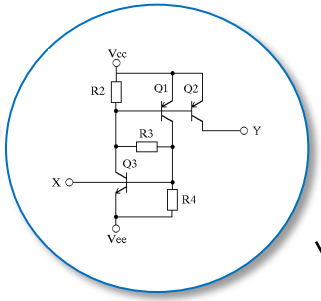
easy to lose time and motivation.



Collective Mind: my wish since 1999

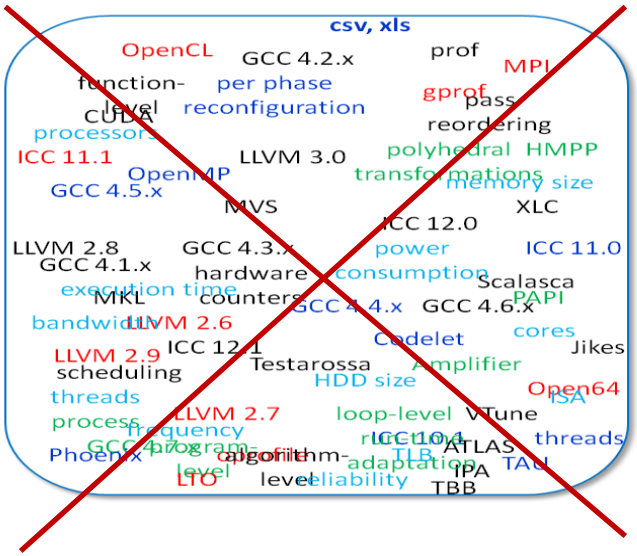
New idea?





Current non-systematic, tedious R&D:

easy to lose time and motivation.



Collective Mind Framework

systematize, unify, share, validate experiments and knowledge

Community

- *download experiment and all cM deps*
- *reproduce / validate*
- *improve*
- *rank*
- *use for teaching*

Academia / Industry

cTuning.org
c-mind.org

Sharing using unified cM web services

Customized repository

Industry

Applications
Benchmarks
Codelets

Datasets

Tools

...

Experimental pipeline

Experimental exploration

Results

Statistical analysis

Models

...

Collective Mind: benefits of common infrastructure

• Researchers can quickly reproduce and validate existing results, and focus their effort on novel approaches combined with data mining, classification and predictive modeling

• Developers can produce tools immediately compatible with collective methodology and infrastructure

• Any person can join collaborative effort to build or extend global expert system that uses Collective Knowledge to:

- *quickly identify program and architecture behavior anomalies*
- *suggest better multi-objective program optimizations and hardware configuration for a given user scenario (requirements)*
- *suggest run-time adaptation scenarios (co-design and co-optimization)*
- *eventually enable self-tuning computer systems*

Collective Mind: current status

- Collective Mind: new plugin-based extensible infrastructure and schema-free repository for collaborative and holistic analysis and tuning of computer systems is currently in validation stage with 2 major companies and with HiPEAC community - plant to release in Autumn 2013
- OpenME interface to “open up” compilers, run-time systems and applications for unified fine-grain analysis and tuning (based on our ICI interface from mainline GCC)
- Hundreds of codelets, thousands of data sets, multiple packages prepared for various research scenarios on data mining
- Plugins for online auto-tuning and predictive modelling
- Portability across all major architectures and OS (Linux, Windows, Android)

Google groups:	Web:	Twitter:
ctuning-discussions	http://cTuning.org	*c_tuning*
collective-mind		*grigori_fursin*

Stay tuned!

Acknowledgements

- PhD students and postdocs

  Abdul Memon, Yuriy Kashnikov

- Colleagues from NCAR, USA

  Davide Del Vento and his colleagues/interns

- Colleagues from IBM, CAPS, ARC (Synopsis), Intel, Google, ARM, ST

- Colleagues from Intel (USA)

  David Kuck and David Wong

- cTuning community:

- EU FP6, FP7 program and HiPEAC network of excellence

  http://www.hipeac.net

Collective Mind Repository and Infrastructure

Systematic application and architecture analysis, characterization and optimization through collaborative knowledge discorvery, systematization, sharing and reuse

Thank you for your attention!

Contact: Grigori.Fursin@cTuning.org

http://cTuning.org/lab/people/gfursin

Open repository to share optimization cases and programs

Gradual parameterization and unification of interfaces of computing systems

Modeling and advice system to predict optimizations, architecture designs, run-time adaptation, etc