# cTuning.org

## Collaborative initiative to create open-source repository and tools to share and reuse knowledge about designs and optimizations of computer systems


Grigori Fursin

HiPEAC COMPILATION ARCHITECTURE  Inria

http://fursin.net/research


*ADAPT or EXTINCT: the time to build self-tuning computer systems ranging from mobile devices to exascale data centers and supercomputers!*

### Main challenges (2012-2018)

Continuing innovation in science and technology is vital for our society and requires ever increasing computational resources. However, delivering such resources became intolerably complex, ad-hoc, costly and error prone due to multiple fundamental reasons:

Performance is no longer the main requirement for new computing systems-multiple objectives must be carefully balanced:
*power consumption, reliability, bandwidth, size, response, portability, design time, various costs*

Too many dimensions and choices for code and architecture design and optimization:
*parallelization and data partitioning; scheduling on heterogeneous architectures with NUMA; contention-aware scheduling in data centers; high-level algorithm tuning; multiple compiler optimizations; polyhedral transformations; instruction-level optimizations; run-time phase-aware optimizations; run-time architecture reconfiguration; multiple ISA extensions*

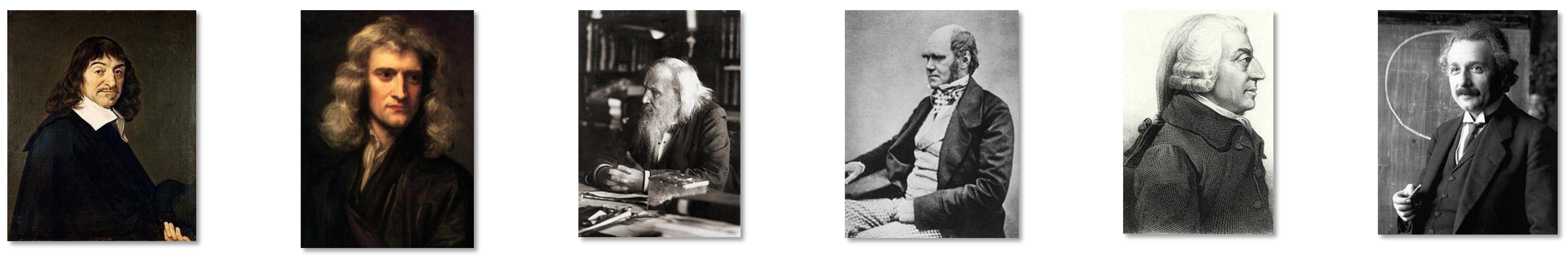
Complex relationships between all components:
*not straightforward to correlate multiple objective functions with multiple design choices and optimizations*


Using outdated, non-adaptive technology results in an enormous waste of expensive computing resources and energy, while considerably slowing down time to market for new technology.

**Current design and optimization methodology has to be dramatically revisited particularly to achieve Exascale performance!**

### cTuning long term interdisciplinary vision

Take the best of existing sciences that deal with complex systems:
computer science, physics, mathematics, chemistry, biology, etc

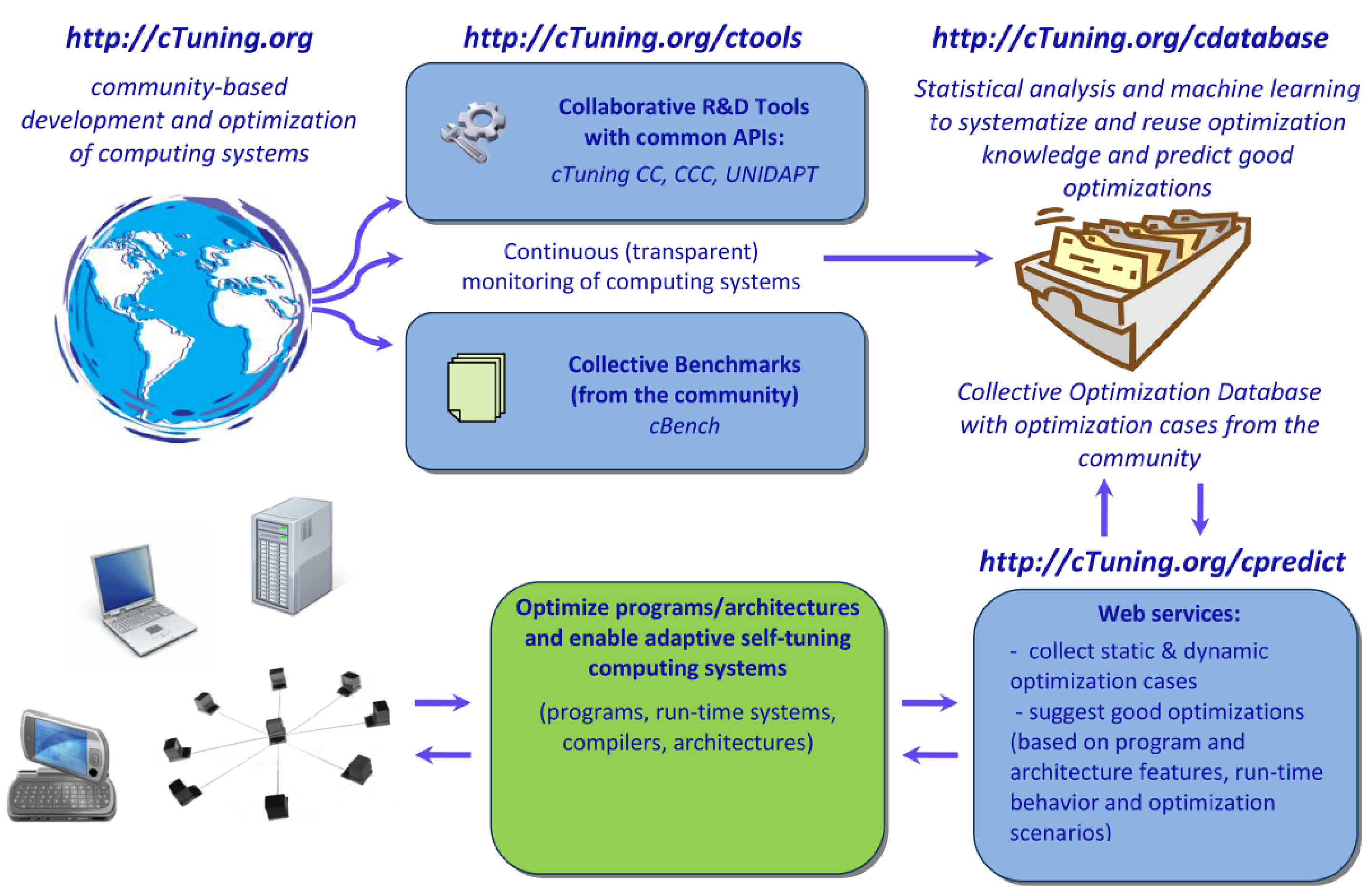


cTuning framework includes methodology, tools, and repository to systematize, quantify, unify and automate architecture and code design, optimization and run-time adaptation based on empirical, analytical and statistical techniques combined with learning, classification, predictive modeling and expert advice web services:

- Extensible and collaborative infrastructure and repository to record the information flow within computer systems
- Continuous data collection and sharing from multiple users
- Collection of unified benchmarks and datasets
- Continuous exploration of multiple design and optimization dimensions
- Data mining techniques to correlate program and architecture features and optimizations
- Embeddable web-services to suggest program optimizations or architecture designs
- Possibility to share and reproduce experimental results (particularly for academic publications)

*Major publications available at http://fursin.net/research and include IJPP'11, ACM TACO'10, PLDI'10, CASES'10, HiPEAC'10, MICRO'09, HiPEAC'09, CGO'07, HiPEAC'07, CGO'06, HiPEAC'05, PhD thesis'04, LCPC'02, CPC'01, etc.*

### Current public version of cTuning infrastructure

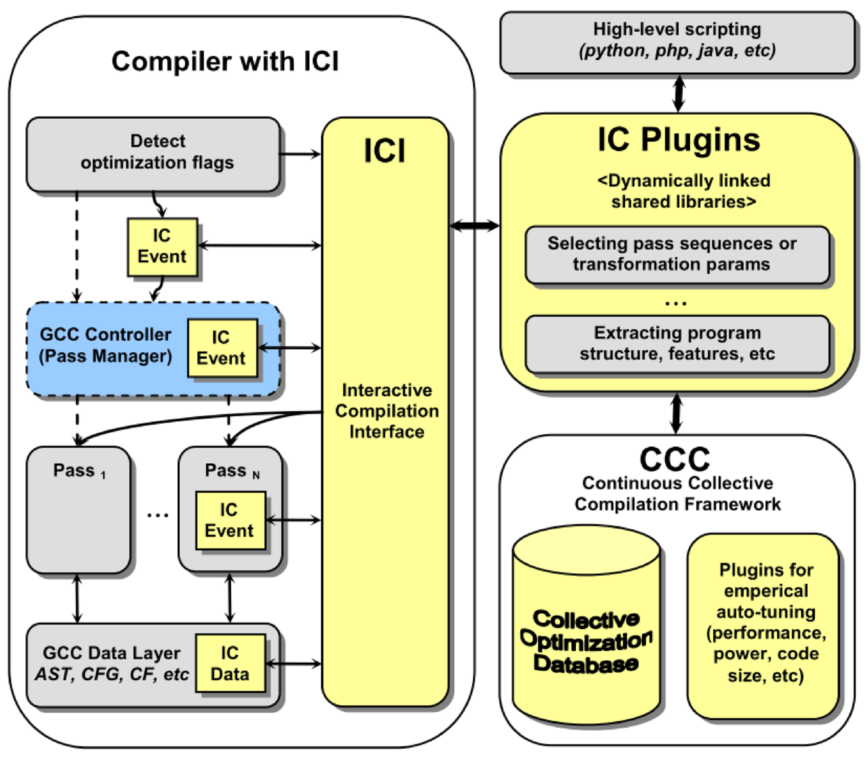



First proof-of-concept version of cTuning infrastructure has been actively used in EU FP6 MHAOTEU project and released in 2009. It includes several empirical auto-tuning tools and machine learning compiler (cTuning CC and MILEPOST GCC) connected to online repository through unified interfaces. cTuning infrastructure has been used and extended by industry and academia. It is currently undergoing major upgrade, following Grigori Fursin's return from industrial sabbatical at Intel Exascale Lab in 2012.

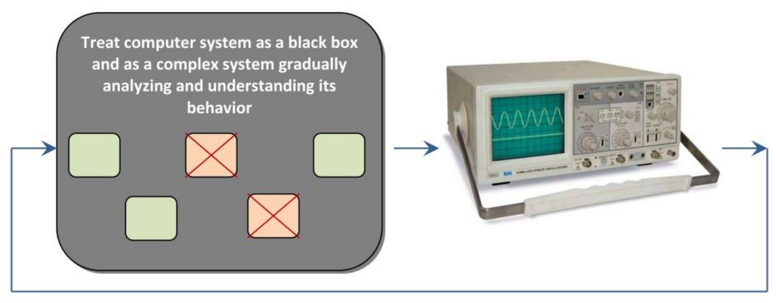
### Empirical analysis and auto-tuning using Interactive Compilation Interface (ICI)

Instead of building new source-to-source or binary-to-binary analysis and optimization infrastructure from scratch, we proposed to "open up" and reuse existing production compilers (Open64 and GCC) and tools using light-weight event-based plugin framework.

*In 2010, ICI has been added to the mainline GCC. We currently investigate possibilities to open up LLVM.*



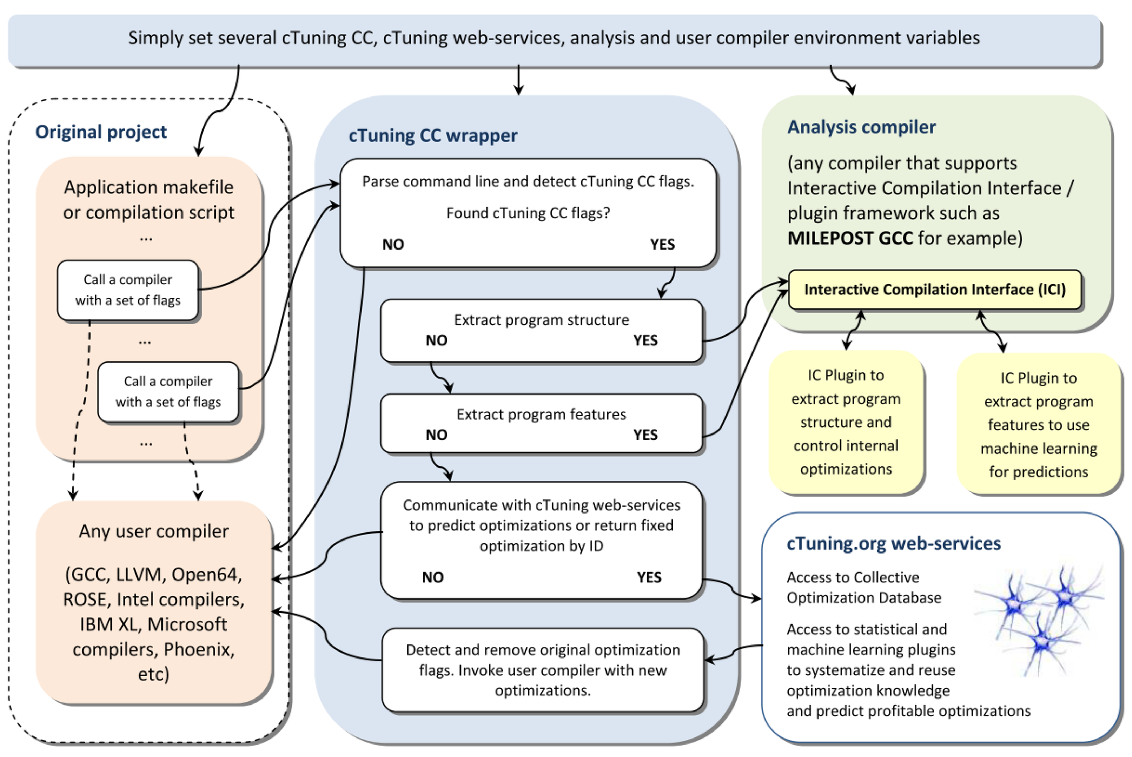

ICI enables transparent for end-users empirical multi-objective auto-tuning (exploration of large optimization spaces) and extraction of program features for further correlation using machine learning techniques.




We developed novel concept to statistically characterize programs and architectures similar to physics through reactions to optimizations or even semantically-non equivalent code modifications

*(removing or adding individual instructions or code segments: for example to detect memory and cache bottlenecks or contentions – see our publications for more details)*

### Machine learning based compilers (cTuning CC/MILEPOST GCC)

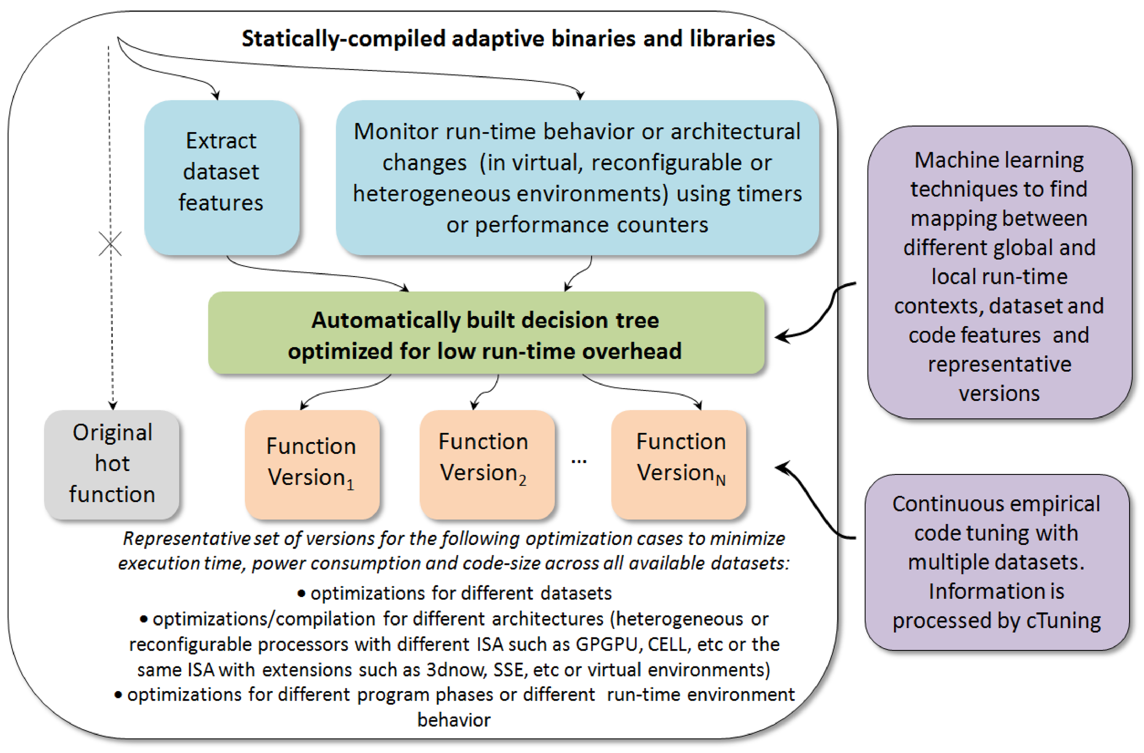



Collecting data from multiple users in a unified way allowed us to continuously apply various data mining (machine learning) techniques to correlate program and architecture behaviour, static and dynamic features, designs and optimizations.

We use continuously updated predictive models (accessible through online web-services) to quickly suggest better optimizations for a given user program, dataset and architecture to balance multiple objectives such as performance, power, compilation time, code size…

First proof-of-concept machine learning compiler connected with cTuning database through unified web-services has been released in 2009 (see IBM's world wide press release "*World's First Intelligent, Open Source Compiler Provides Automated Advice on Software Code Optimization*") . Since then, it has been extended within multiple collaborative projects and Google Summer of Code program. In 2012, cTuning CC will be undergoing major upgrade. More info can be found at http://cTuning.org/ctuning-cc

### Machine learning based run-time adaptation for self-tuning computing systems




Using statistical and machine learning techniques on the continously collected data allows to detect representative sets of optimizations that cover varying program behavior due to different datasets, run-time local and system behavior, etc.

Combined with UNIDAPT framework (see HiPEAC'05, TACO'10), we can now create self-tuning binaries and libraries that can automatically select appropriate optimizations or reconfigure architectures as a reaction to different program behavior, architectural changes, contentions, etc.

Proof-of-concept version of UNIDAPT framework for run-time adaptation has been extended within several collaborative projects funded by HIPEAC and ICT, and within Google Summer of Code ("Predictive runtime code scheduling for heterogeneous architectures", "Collective Optimization: A Practical Collaborative Approach"). In 2012, UNIDAPT framework will be undergoing major upgrade. More info can be found at http://cTuning.org/unidapt

### Current and future work (contact Grigori Fursin for more details):

Redesigning cTuning based on user feedback and new research ideas

*Major upgrade with a focus on (heterogeneous) multicore systems;*
*Open source release of cTuning$_2$ framework in expected in summer 2012;*
*Sponsorship and industrial support are very welcome!*

### Contact and further information: **Grigori Fursin (cTuning founder and R&D leader)**


*grigori.fursin@inria.fr*


*http://fursin.net/research* (publications, projects, presentations)

*http://cTuning.org* (tools, benchmarks, datasets, web-services, collaborative wiki)
*http://groups.google.com/group/ctuning-discussions* (public discussions)