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.
- Too many dimensions and choices for computer systems design and optimization.
- Complex dependencies between multiple objective functions and design choices.
- Heterogeneity of computer systems and platforms.
- Continuously increasing size and complexity of software.
- Difficulty in correlating multiple objective functions with multiple design choices and optimizations.
- Intolerance of computer systems to non-equivalent code modifications.
- Hoc, costly and error prone due to many complicated factors.

Current design and optimization methodology has to be dramatically revisited particularly to achieve expressable performance.

Current public version of cTuning infrastructure

http://cTuning.org

- Community-based development and optimization of computing systems
- Socialization and sharing of computer systems and optimizations
- Automatic, transparent, unifying compilation tools and services
- Statistical and machine learning techniques to correlate program and architecture features and optimizations
- Continuous data collection and sharing from multiple users
- Extensible and collaborative infrastructure and repository to record the information flow within tuning computer systems ranging from mobile devices to exascale data centers and supercomputers.

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, computation time, code size...

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

Using statistical and machine learning techniques on the continuously collected data allows to detect representative sets of optimizations that cover varying program behavior due to different datasets, run-time and system behavior. By combining UNIDAPT framework (see HPEC/AC'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, contents, etc.

Machine learning framework includes methodology, tools, and repository to:

- Systematize, quantify and automate architecture and code design, optimization and run-time adaptation based on empirical, analytical and statistical techniques.
- Support computer science, physics, mathematics, chemistry, biology, etc.

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 heterogeneity in multicore systems
- Open source release of cTuning, 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.info
http://fursin.net/research
http://cTuning.org (tools, benchmarks, web-services, collaborative wiki)
http://groups.google.com/group/ctuning-discussions (public discussions)

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 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 contensions – see our publications for more details).