cTuning.org
novel extensible methodology, framework
and public repository to collaboratively
address Exascale challenges

Path towards Exascale computing (2012-2018)

Which path to choose?

Quick, non-reproducible hack?
Arbitrarily heuristic? Quick publication?

Long term optimization and tuning of large and complex computer systems through continuous and interactive online compilers and tools

End-user task

Collaborative infrastructure and repository

Common to all presenting solutions

Major challenges:

• Too many design and optimization choices
• Finding the right solution is extremely challenging
• Slow time to market, low ROI
• Multiple user objectives vs. choices; benefit vs. costs
• Too many papers with non representative results: technological chaos
• Too many papers with non-reproducible results: technological chaos

Consequences:

• Finding the right solution is extremely challenging and time consuming: everyone is lost in choices
• Slow time to market, low ROI
• Underperforming systems wasting expensive energy
• Attracting new students is difficult: bad image of ad-hoc solutions and hacking
• Sharing innovation in science and technology

Collective Mind infrastructure and repository for online auto-tuning and learning

ICl Plugins

ICl Plugins

Mainline GCC

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 information flow within computer systems
• Continuous data collection and sharing from multiple users
• Collection of unified benchmarks, codelets, micro-benchmarks and datasets
• Continuous exploration of multiple design and optimization dimensions
• Plugins for online data mining and machine learning techniques to extrapolate existing knowledge to build faster, more powerful and efficient devices
• Public web-services to suggest optimal program optimizations or architecture designs
• New publication model to reproduce experimental results by the community

Empirical analysis and auto-tuning using interactive compilers

Novel concept to convert rigid tools into powerful interactive toolsets: using light-weight event-based plugin framework.

We developed Interactive Compilation Interface (ICI) to “open up” existing compilers such as Open64 and GCC, and to enable transparent for end-users empirical multi-objective auto-tuning on the fly (exploration of large optimization spaces) and extraction of program features to enable predictive modelling.

In 2010, ICI (plugin framework) was added to the mainline GCC. We are developing a new version of the plugin framework (“Open64ICL”) to be able to open up any existing compiler or tool.

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

(removing or adding individual instructions, code segments, threads, etc. for example to detect memory and cache bottlenecks or contentions)

Building self-tuning computer systems

(combining static and dynamic approaches with online learning)

We developed a novel approach to statistically enable dynamic optimizations (UNDAPT framework) by combining a small set of pre-optimized versions of a code with online learning plugins to quickly select the most appropriate versions at run-time adapting to a given (heterogeneous) architecture or varying program phase due to multiple datasets, contentions, etc.

Collective Mind framework uses machine learning techniques to continuously determining minimal representative sets of codelets and optimizations that cover varying program behavior due to different datasets, run-time program and system behavior, etc.