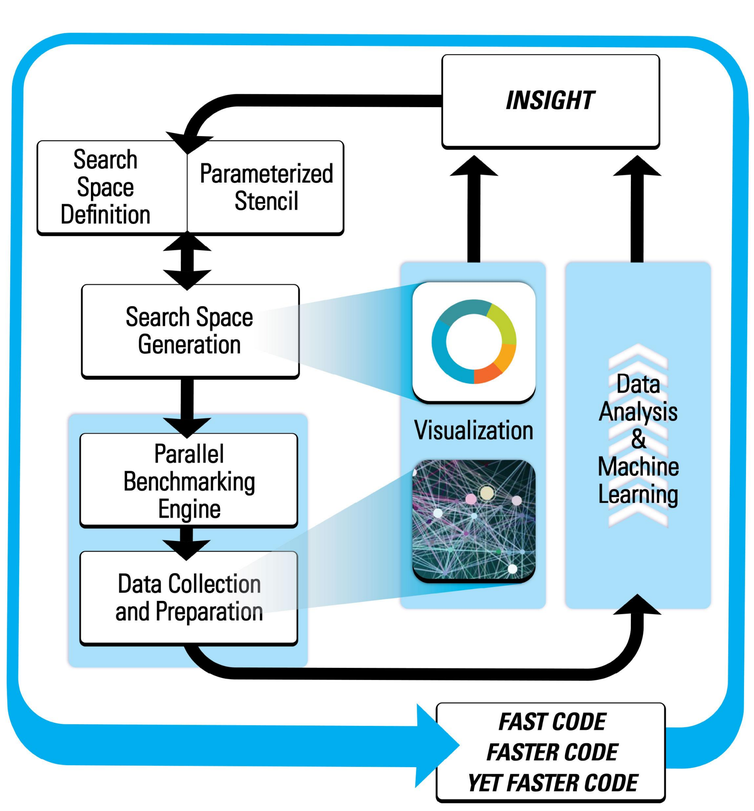
BONSAI (Benchtesting OpeN Software Autotuning Infrastructure)

Yaohung Mike Tsai, Matthew Bachstein, Piotr Luszczek, Jakub Kurzak, Hartwig Anzt, Mark Gates, Jack Dongarra
INNOVATIVE COMPUTING LABORATORY, UNIVERSITY OF TENNESSEE

The goal of the BONSAI (Benchtesting OpeN Software Autotuning Infrastructure) project is to develop a software infrastructure for using parallel hybrid systems at any scale to carry out large, concurrent autotuning sweeps in order to dramatically accelerate the optimization process of computational kernels for GPU accelerators and many-core coprocessors.



Overview

These are the core components of BONSAI, aiming to provide an infrastructure for kernel developers to rapidly design search space and test through massive data sets and data layouts.

LANAI
LANguage for Autotuning Infrastructure

LANAI is a Python-based language for specifying the search space and pruning constraints for autotuning kernels. The search space is defined by **iterators**. An **iterator** can be an expression or deferred from other **iterators**. Combined with **conditions** to remove undesired cases, LANAI would automatically generate the search space by applying Cartesian product over **iterators** with optimal order. In addition for NVIDIA CUDA kernels, LANAI has built in constants like *MaxRegistersPerThread* for developers to take care of hardware constraints according to the architecture of the GPU being used.

```
R1 = range(3)
R2 = range(3)

@condition
def r1_smaller_than_r2():
    return R1 < R2
```

```
R1, R2
0 , 0
1 , 0
1 , 1
2 , 0
2 , 1
2 , 2
```

BONSAI API

In order to use BONSAI, the user has to divide their program into three parts to fit into provided skeleton: (1) **Initialization**, setting up all variables and data in memory for the kernel, (2) **Execution**, the actual kernel call for the main computation kernel, and (3) **Finalization**, correctness check and clean up the memory. By default, BONSAI will measure the time needed to execute the kernel. Additional profiling counters can be enabled through the BONSAI API for more detailed analysis. Also, the user can provide other information like numerical error by API calls. The result is gathered automatically to generate the autotuning sweep report.

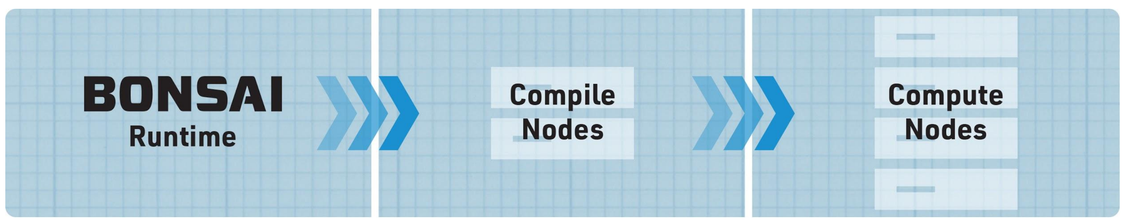
BONSAI RUNTIME

The BONSAI Runtime takes care of the parallel compilation and execution, as well as file/memory management. BONSAI will try to use all available GPUs/Accelerators to speed up the tuning process. User does not need to handle the job distribution or memory allocation for multiple GPUs. This frees users from implementing their own MPI+OpenMP/CUDA/OpenCL tuning program and greatly simplifies the compilation process.

Key Features

PARALLEL COMPILATION AND EXECUTION ON CLUSTER

From a single node with multiple GPUs to complicated clusters with separated login/complication and computation nodes, BONSAI provides the infrastructure to do parallel tuning sweeps on all kind of systems. Users do not need to program for multi-GPU or MPI communication. BONSAI automatically distributes all the tuning jobs among available resources. BONSAI not only reduces the time to perform the autotuining sweep, but also the time programmers spend to set up the tuning work.



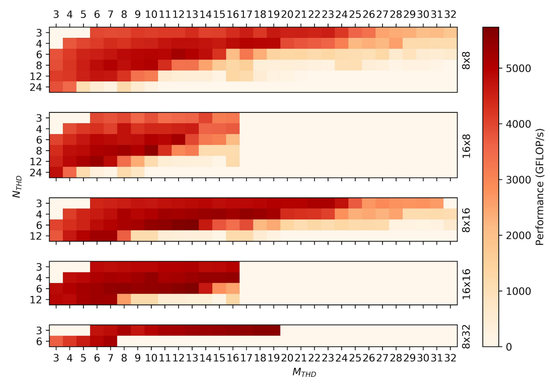
PROFILING
IN DEVELOPMENT

BONSAI is integrated with profiling libraries/tools such as PAPI and CUDA Profiler. BONSAI simplifies the task of instrumenting the kernel and provides a simple interface for selecting counters to be collected. BONSAI will gather all the information from distributed executions. It creates the opportunity for developers to learn more insights about their kernel and the possibility to optimize it further.

STATISTICAL ANALYSIS AND VISUALIZATION
IN DEVELOPMENT

We will provide a number of analytical tools and examples to guide the developer in analyzing their code. The analytical tools provided with BONSAI will include statistical and machine learning tools in addition to a number of visualization utilities. These tools will leverage open-source data analysis libraries such as the SciPy stack, R, and Spark based MLlib.

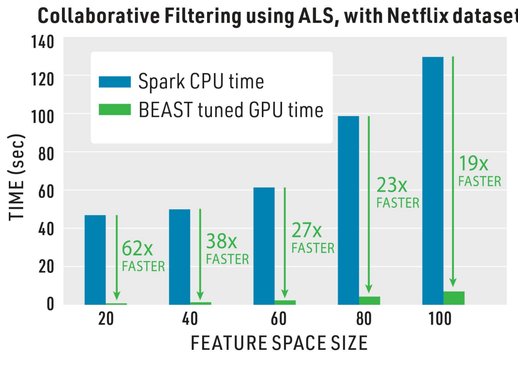
Visualization for Tuning Result of Convolution Layers from Deep Learning



Y. M. Tsai, P. Luszczek, J. Kurzak, J. Dongarra
Performance-portable Autotuning of Opencl Kernels for Convolutional Layers of Deep Neural Networks
2016 2nd Workshop on Machine Learning in HPC Environments (MLHPC), Salt Lake City, UT, 2016, pp. 9-18. DOI: 10.1109/MLHPC.2016.005

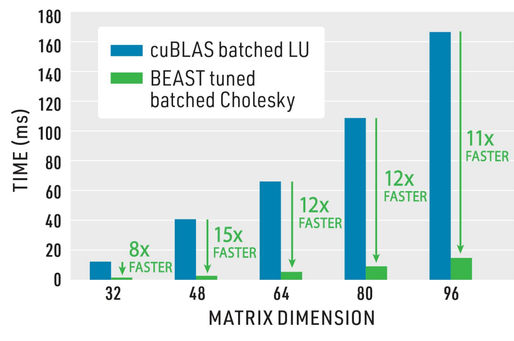
Tuning Works

Collaborative Filtering using ALS, with Netflix dataset



M. Gates, H. Anzt, J. Kurzak, J. Dongarra
Accelerating Collaborative Filtering Using Concepts from High Performance Computing
BigData'15: IEEE International Conference on Big Data, Santa Clara, CA, 2015.
DOI: 10.1109/BigData.2015.7363811

Batched Matrix Factorization



M. Gates, H. Anzt, J. Kurzak, J. Dongarra
Implementation and Tuning of Batched Cholesky Factorization and Solve for NVIDIA GPUs
Transactions on Parallel and Distributed Systems, 27(7):2036-2048, 2015.
DOI 10.1109/TPDS.2015.2481890

SPONSORED BY NSF

ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

FIND OUT MORE AT
https://bitbucket.org/icl/bonsai