Autotuning Batched Kernels:

- Batched Cholesky with Interleaved Layout
- The BONSAI Project

J. Kurzak (PI)
P. Luszczek (PI)
M. Gates
Y. Tsai
M. Bachstein
Y. Pei
Batched Cholesky
interleaved layouts
Batched Cholesky
interleaved layouts
Batched Cholesky
algorithmic blocking

\[ N \]

\[ N \]

\[ NB \]

\[ POTRF \]

\[ SYRK \]

\[ TRSM \]
Batched Cholesky
algorithmic variants
Batched Cholesky tiling

- SYRK
- POTRF
- GEMM
- TRSM
Batched Cholesky tuning parameters

**Algorithmic Variant**
- right-looking
- left-looking
- top-looking

**Unrolling**
- partial (tile operation only)
- full (the entire routine)

**Tiling Factor (NB)**

**Chunking**
- no chunking
- chunking
  - chunking factor (DIM)

**target:** NVIDIA Pascal GPU  
span sizes up to 100  
~14,000 runs
Batched Cholesky
overall performance
Batched Cholesky
overall performance

![Graph showing speedup over magma vs matrix dimension (n) for interleaved and interleaved, fast math cases.](image)
Batched Cholesky tiling
Batched Cholesky
algorithmic variants

![Graph showing performance of different algorithms against matrix dimension. The graph plots Gflop/s on the y-axis against matrix dimension (n) on the x-axis. There are four lines representing different algorithms: 'top', 'left', 'right', and 'magma'. The 'top' algorithm shows the highest performance, followed by 'left', 'right', and 'magma' which has the lowest performance.]
Batched Cholesky
chunking
Batched Cholesky
chunk size

![Graph showing the performance of Batched Cholesky for different chunk sizes. The graph plots Gflop/s against matrix dimension (n). The performance is compared for different chunk sizes: size 32, size 64, size 128, size 256, size 512, and magma.]
Batched Cholesky unrolling
BEAST
Bench-testing Environment for Automated Software Tuning

- generate large search space
- prune in a smart way
- collect results
- iterate
BEAST
traditional batched Cholesky

Kurzak, Jakub, Hartwig Anzt, Mark Gates, and Jack Dongarra.
"Implementation and tuning of batched Cholesky factorization and solve for NVIDIA GPUs."
*IEEE Transactions on Parallel and Distributed Systems* 27, no. 7 (2016): 2036-2048.
BEAST
LANAI: LANguage for Autotuning Infrastructure

```python
dim_m = range(1, max_threads_dim_x+1)
dim_n = range(1, max_threads_dim_y+1)
@iterator
def blk_m(dim_m):
    return range(dim_m, max_threads_dim_x+1, dim_m)
@iterator
def blk_n(dim_n):
    return range(dim_n, max_threads_dim_y+1, dim_n)
blk_k = range(1, min(max_threads_dim_x, max_threads_dim_y)+1)
@iterator
def dim_vec(arithmetic, precision):
    if arithmetic == "double":
        if precision == "real":
            return range(1, 3)
    else:
        return 1
else:
    if precision == "real":
        return range(1, 5, 3)
    else:
        return range(1, 3)
```

Luszczek, Piotr, Mark Gates, Jakub Kurzak, Anthony Danalis, and Jack Dongarra.
"Search space generation and pruning system for autotuners."
BONSAI
BEAST OpenN Software Autotuning Infrastructure

- deploy large autotuning sweeps to supercomputers or the Cloud
- distributed compilation
- distributed benchmarking
- dynamic load balancing
- exhaustive collection of performance data
- postprocessing of performance data
- data analysis
- visualization