Cray Scientific Libraries

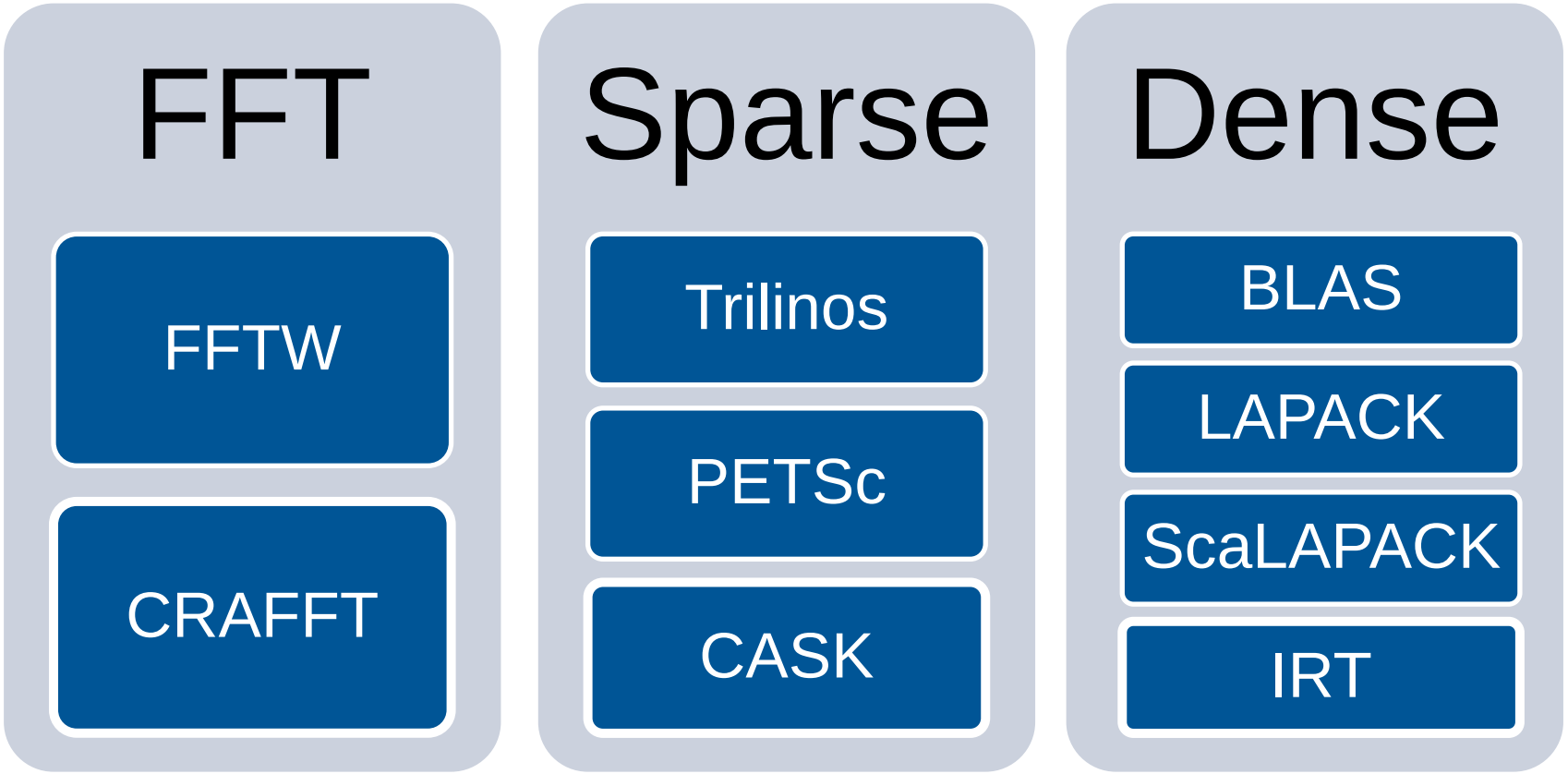
Overview

What are libraries for?

- **Building blocks for writing scientific applications**
- **Historically – allowed the first forms of code re-use**
- **Later – became ways of running optimized code**
- **Today the complexity of the hardware is very high**
- **The Cray PE insulates users from this complexity**
  - Cray module environment
  - CCE
  - Performance tools
  - Tuned MPI libraries (+PGAS)
  - Optimized Scientific libraries

Cray Scientific Libraries are designed to provide the maximum possible performance from Cray systems with minimum effort.

Scientific libraries on XC – functional view



What makes Cray libraries special

1. **Node performance**
   - Highly tuned routines at the low-level (ex. BLAS)
2. **Network performance**
   - Optimized for network performance
   - Overlap between communication and computation
   - Use the best available low-level mechanism
   - Use adaptive parallel algorithms
3. **Highly adaptive software**
   - Use auto-tuning and adaptation to give the user the known best (or very good) codes at runtime
4. **Productivity features**
   - Simple interfaces into complex software

LibSci usage

- **LibSci**
  - The drivers should do it all for you – no need to explicitly link
  - For threads, set OMP_NUM_THREADS
    - Threading is used within LibSci
    - If you call within a parallel region, single thread used
- **FFTW**
  - `module load fftw` (there are also wisdom files available)
- **PETSc**
  - `module load petsc` (or module load petsc-complex)
  - Use as you would your normal PETSc build
- **Trilinos**
  - `module load trilinos`
- **Cray Adaptive Sparse Kernels (CASK)**
  - You get optimizations for free

Your friends

- module command (module --help)
- PrgEnv modules:
- Component modules
- csmlversion (tool)
- Cray driver scripts ftn, cc, CC

```
TUNER/STUNER> module avail PrgEnv

PrgEnv-cray/3.1.35          PrgEnv-gnu/4.0.12A          PrgEnv-
pathscale/3.1.37G
PrgEnv-cray/3.1.37AA        PrgEnv-gnu/4.0.26A          PrgEnv-
pathscale/3.1.49A
PrgEnv-cray/3.1.37C         PrgEnv-gnu/4.0.36(default)  PrgEnv-
pathscale/3.1.61
PrgEnv-cray/3.1.37E         PrgEnv-intel/3.1.35         PrgEnv-
pathscale/4.0.12A
PrgEnv-cray/3.1.37G         PrgEnv-intel/3.1.37AA       PrgEnv-
pathscale/4.0.26A
PrgEnv-cray/3.1.49A         PrgEnv-intel/3.1.37C        PrgEnv-
pathscale/4.0.36(default)
PrgEnv-cray/3.1.61          PrgEnv-intel/3.1.37E        PrgEnv-pgi/3.1.35
PrgEnv-cray/4.0.12A         PrgEnv-intel/3.1.37G        PrgEnv-
pgi/3.1.37AA
PrgEnv-cray/4.0.26A         PrgEnv-intel/3.1.49A        PrgEnv-pgi/3.1.37C
PrgEnv-cray/4.0.36(default) PrgEnv-intel/3.1.61         PrgEnv-
pgi/3.1.37E
PrgEnv-gnu/3.1.35           PrgEnv-intel/4.0.12A        PrgEnv-pgi/3.1.37G
PrgEnv-gnu/3.1.37AA         PrgEnv-intel/4.0.26A        PrgEnv-
pgi/3.1.49A
PrgEnv-gnu/3.1.37C          PrgEnv-intel/4.0.36(default) PrgEnv-
pgi/3.1.61
PrgEnv-gnu/3.1.37E          PrgEnv-pathscale/3.1.35     PrgEnv-
pgi/4.0.12A
PrgEnv-gnu/3.1.37G          PrgEnv-pathscale/3.1.37AA   PrgEnv-
pgi/4.0.26A
PrgEnv-gnu/3.1.49A          PrgEnv-pathscale/3.1.37C    PrgEnv-
pgi/4.0.36(default)
PrgEnv-gnu/3.1.61           PrgEnv-pathscale/3.1.37E
```

```
------------------------------------------------ /opt/cray/modulefiles ------------------------------------
-------

xt-libsci/10.5.02           xt-libsci/11.0.04           xt-libsci/11.0.05.1
xt-libsci/11.0.03           xt-libsci/11.0.04.8         xt-libsci/11.0.05.2(default)
```

Check you got the right library!

- Add options to the linker to make sure you have the correct library loaded.
- `-Wl` adds a command to the linker from the driver
- You can ask for the linker to tell you where an object was resolved from using the –y option.
  - E.g. `–Wl, -ydgemm_`

```
.//main.o: reference to dgemm_
/opt/xt-libsci/11.0.05.2/cray/73/mc12/lib/libsci_cray_mp.a(dgemm.o):
definition  of dgemm_
```

Note: do not explicitly link “-lsci”. This will not be found from libsci 11+ and means a single core library for 10.x.

Threading

- **LibSci is compatible with OpenMP**
  - Control the number of threads to be used in your program using **OMP_NUM_THREADS**
  - e.g., in job script **`export OMP_NUM_THREADS=16`**
  - Then run with **`aprun -n1 -d16`**
- **What behavior you get from the library depends on your code**
  1. No threading in code
     - The BLAS call will use OMP_NUM_THREADS threads
  2. Threaded code, outside parallel regions
     - The BLAS call will use OMP_NUM_THREADS threads
  3. Threaded code, inside parallel regions
     - The BLAS call will use a single thread

Threaded LAPACK

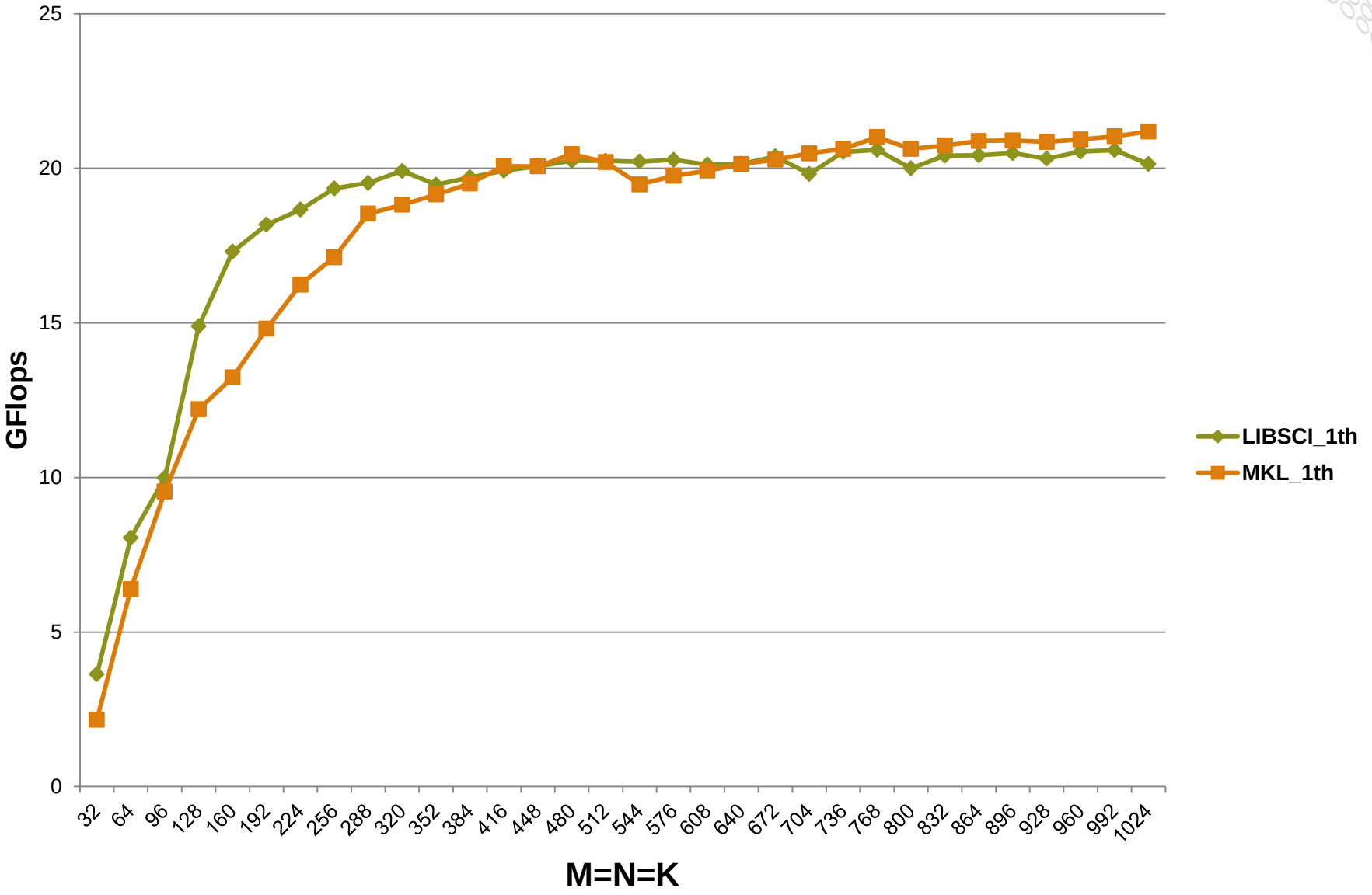
- Threaded LAPACK works exactly the same as threaded BLAS
- Anywhere LAPACK uses BLAS, those BLAS can be threaded
- Some LAPACK routines are threaded at the higher level
- No special instructions

Performance Focus and Autotuning

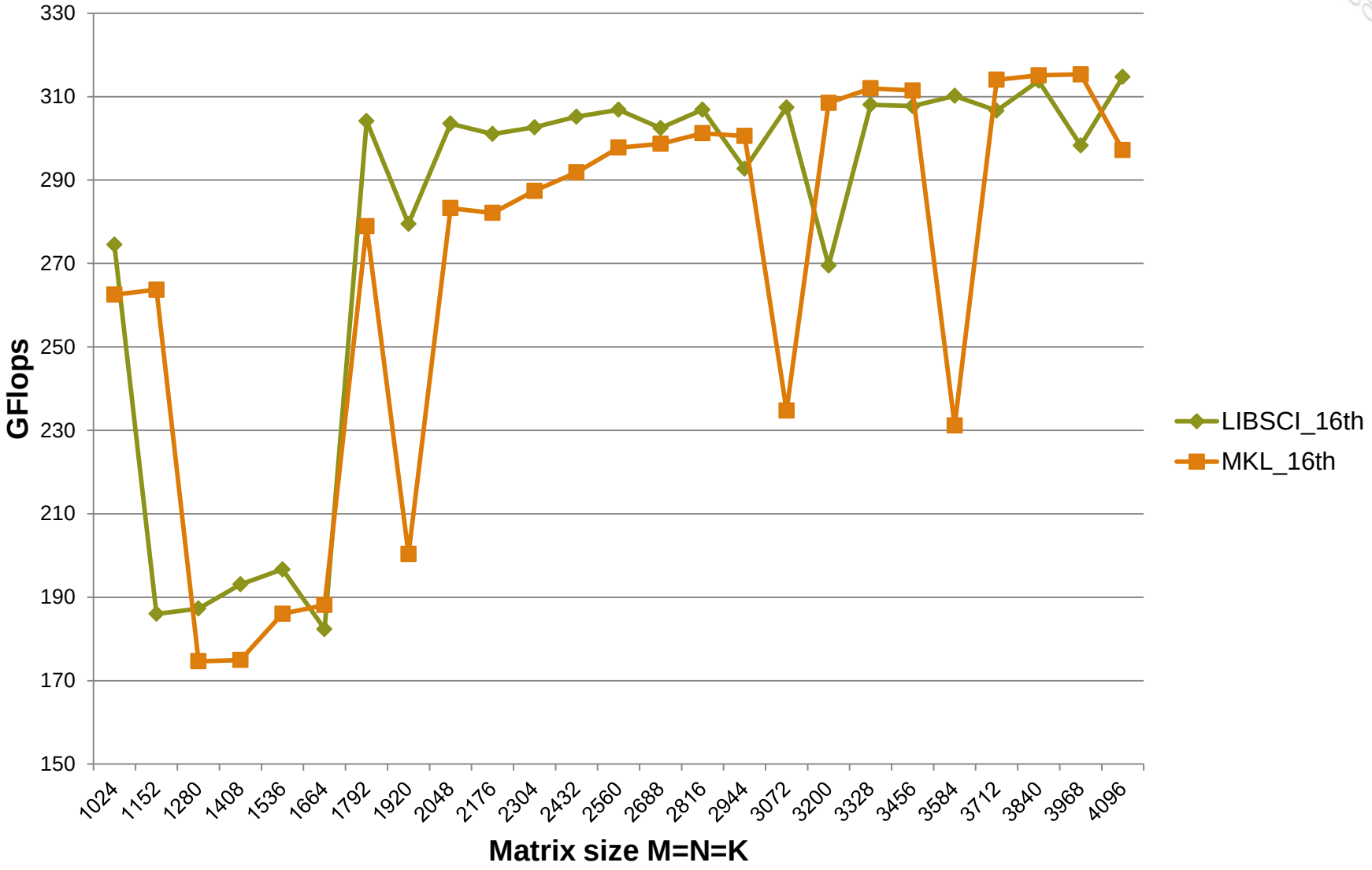
- **Some components of the library are performance critical**
  - For example BLAS and specifically GEMM
- **It is a significant challenge to get best performance across a range of architectures and problem sizes and thread counts**

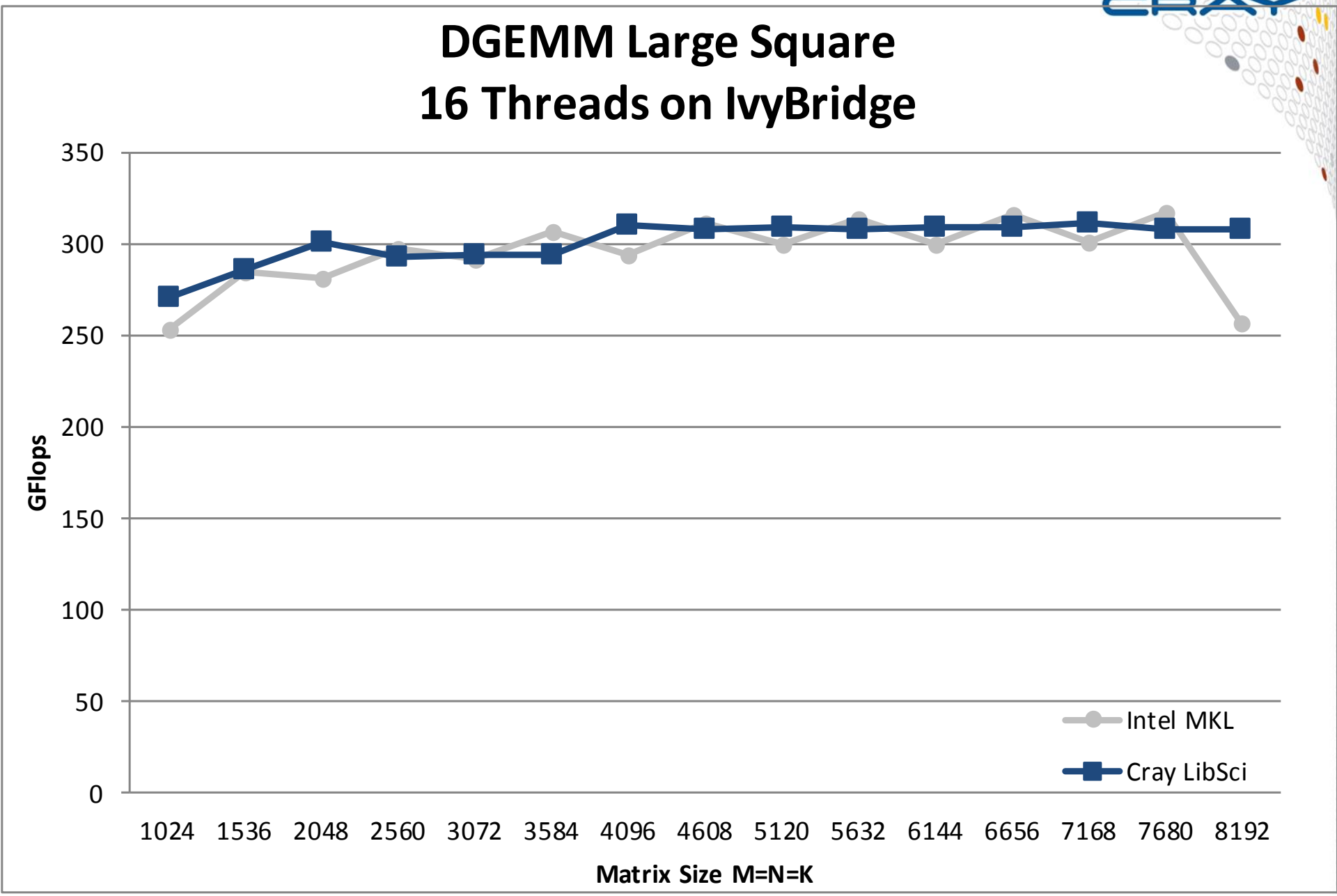
- **Cray has an autotuning framework to address this:**
  - It uses a general GEMM framework
  - Offline tuning runs are done for a wide range of problem sizes
  - CPU and GPU targets
  - Knowledge gained from offline runs incorporated into the runtime library.

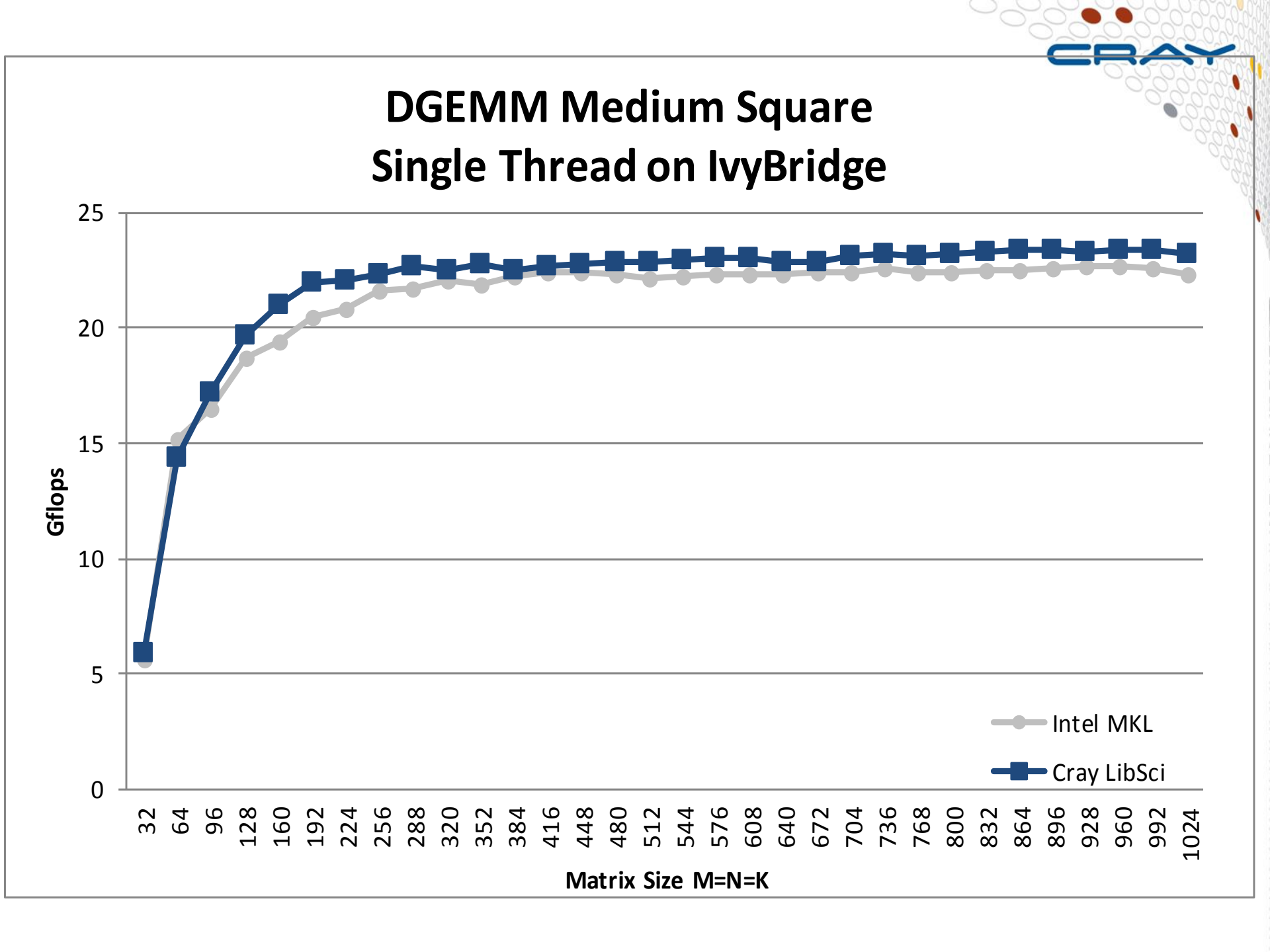
SANDYBRIDGE DGEMM - MEDIUM SQUARE
LIBSCI vs MKL (single thread)



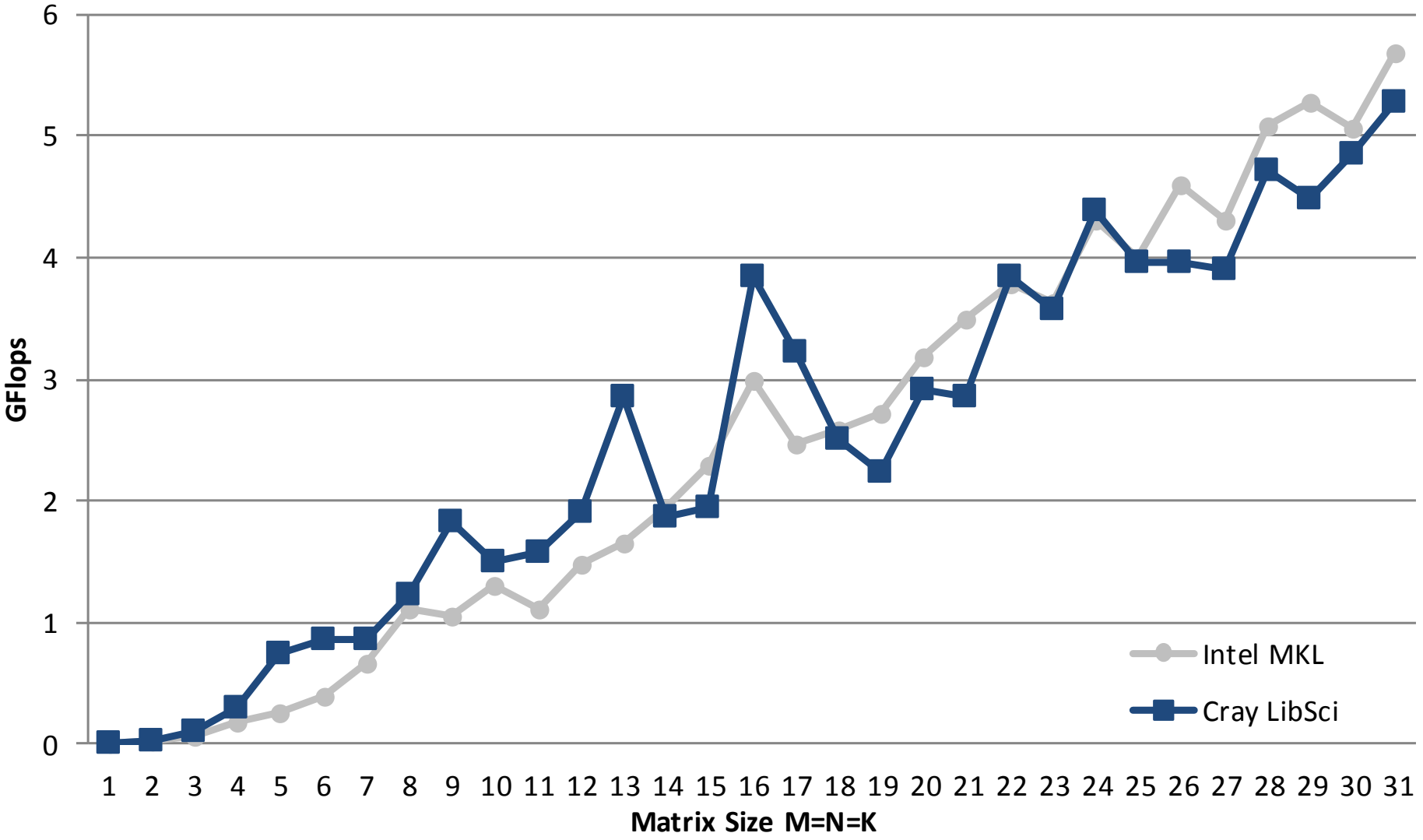
SANDYBRIDGE DGEMM - LARGE SQUARE
LIBSCI vs MKL



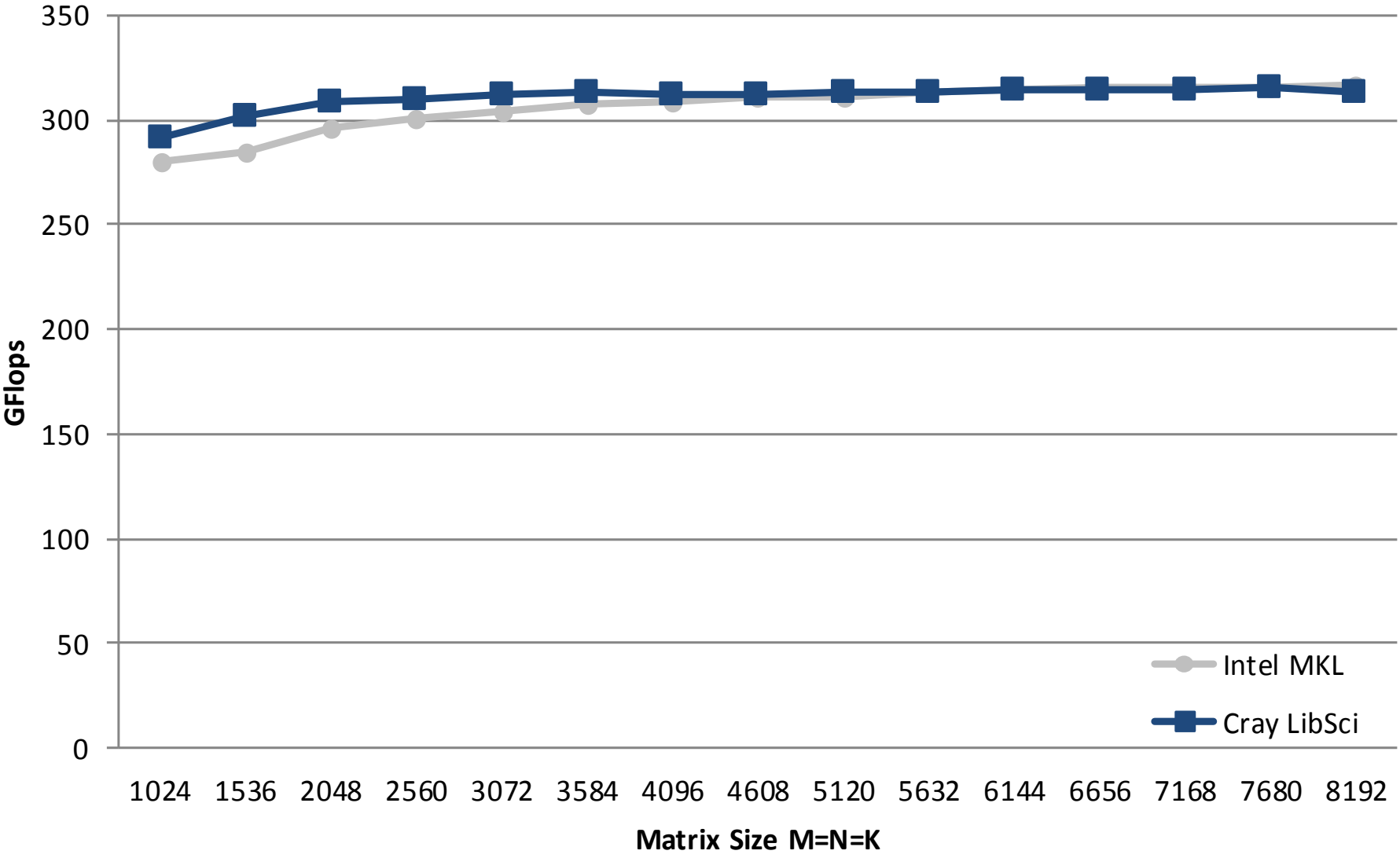

DGEMM Large Square
16 Threads on IvyBridge
GFlops
350
300
250
200
150
100
50
0
1024
1536
2048
2560
3072
3584
4096
4608
5120
5632
6144
6656
7168
7680
8192
Matrix Size M=N=K
Intel MKL
Cray LibSci


DGEMM Medium Square
Single Thread on IvyBridge
Gflops
25
20
15
10
5
0
32
64
96
128
160
192
224
256
288
320
352
384
416
448
480
512
544
576
608
640
672
704
736
768
800
832
864
896
928
960
992
1024
Matrix Size M=N=K
Intel MKL
Cray LibSci

DGEMM Tiny Square
Single Thread on IvyBridge



ZGEMM Large Square
16 Threads on IvyBridge



Tuning requests

- **CrayBLAS is an auto-tuned library**
  - Generally, excellent performance is possible for all shapes and sizes
- **However, the adaptive CrayBLAS can be improved by tuning for exact sizes and shapes**
- **Send your specific tuning requirements to crayblas@cray.com**
  - Send the routine name and the list of calling sequences

ScaLAPACK and IRT

- **ScaLAPACK in LibSci is optimized for Gemini/Aries interconnect**
  - New collective communication procedures are added
  - Default topologies are changed to use the new optimizations
  - Much better strong scaling
- **It also benefits from the optimizations in CrayBLAS**
- **Iterative Refinement Toolkit (IRT) can provide further improvements**
  - Uses mixed precision
  - For some targets (CPU vector instructions and GPUs) single-precision can be much faster
  - Used for serial and parallel LU, Cholesky and QR
  - Either set IRT_USE_SOLVERS to 1
    or use the advanced API.

Cray Adaptive FFT (CRAFFT)

- **Serial version really just a productivity enhancer**
- **Supports plan/execute with wisdom or do both at once**
- **Load the module**
- **Fortran: use crafft**
- **Serial:**
  - Various simple-to-use interfaces with optional arguments
- **Parallel:**
  - Provides efficient network transposes but uses FFTW3 serial transforms
  - Various network optimizations including computation and communication overlap
  - Various 2d/3d real and complex transforms implemented

Summary

- **Do not re-invent the wheel but use scientific libraries wherever you can!**
- **All the most widely used library families and frameworks readily available as XE/XC optimized versions**
  - And if the cornerstone library of your application is still missing, let us know about it!
- **Make sure you use the optimized version provided by the system instead of a reference implementation**

- **... and give us feedback**