CGT 581I - Parallel Graphics and Simulation

Performance Libraries

Bedrich Benes, Ph.D.
Professor
Department of Computer Graphics
Purdue University

Intel Performance Libraries

- Three libraries MKL, DAAL, and IPP

1) **Intel Math Kernel Library (MKL)**
- Basic Linear Algebra Subprograms (BLAS)
- Fast Fourier Transform (FFT)
- Equation Solver, etc.

Intel Performance Libraries

- Three libraries MKL, DAAL, and IPP

2) **Intel Data Analytics Acceleration Library (DAAL)**
- accelerate big data processing
- use with Spark (cluster computing), Hadoop (distributed storage and processing of big data), etc.

Intel Performance Libraries

- Three libraries MKL, DAAL, and IPP

3) **Intel Integrated Performance Primitives (IPP)**
- image,
- media,
- communication, and
- signal processing

Intel Performance Libraries

- are either downloadable for free
- or distributed as a part of Parallel Studio
- cross Intel Architecture compatible
- new improved versions, good support
- fully supported at Knights Landing (use AVX-512)
- often used in different packages
- also a version of R
- an Anaconda Python distribution

Native or Offload

Model	Description
Native	Functions are called from the application Executed natively on Xeon Phi
Automatic Offload (AO)	MKL capability Some functions are distributed over KNL processor(s) Automatic data flow control C/C++ Fortran
Compiler-Assisted Offload (CAO)	C/C++ pragmas or Fortran directives control A library func is called and runs on Xeon Phi target(s) Support for multiple Xeon Phi(s)

Native or Offload

- offload uses KNL as coprocessor
- can be sent over to a KNL over PCIEx
- Math Kernel Library – AO, CAO
- DAAL – CAO
- IPP – CAO

Intel Math Kernel Library

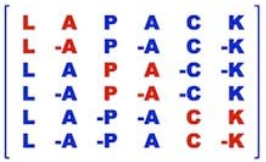
- Runs on Windows, OS X, and Linux
- Fortran, C/C++, Python (NumPy, SciPy) Java, C#
- Cluster support
- Thread Building Blocks (TBB) support

Intel Math Kernel Library

- Includes:
- **Basic Linear Algebra Subprograms (BLAS)**
  - Level 1: vector operations
  - Level 2: matrix-vector operations
  - Level 3: matrix-matrix operations
- **Sparse BLAS**
  - Level 1-3
- **PBLAS**
  - parallel BLAS - distributed

Intel Math Kernel Library

- Includes:
- **Linear Algebra Package (LAPACK)**
  - linear equations solvers
  - least squares
  - eigenvalus
  - singular value problems
  - Sylvester's equation



- **ScaLAPACK**
  - additional routines on a cluster

Intel Math Kernel Library

- Includes:
- **MKL PARDISO**
  - Cholesky
  - LU factorization
  - Direct sparse solver
- **Deep Neural Network Primitives**
- **Extended Eigensolver**
- **Vector Math**
- **Random numbers**
- **FFT, cluster FFT**

Calling BLAS Level 1 Function from C

```
#include <stdio.h>
#include "mkl.h"
#define N 5
int main(){
 int n = N, inca = 1, incb = 1, i;
 MKL_Complex16 a[N], b[N], c;
 for( i = 0; i < n; i++ ){
   a[i].real = (double)i; a[i].imag = (double)i * 2.0;
   b[i].real = (double)(n - i); b[i].imag = (double)i * 2.0;
 }
 zdotc( &c, &n, a, &inca, b, &incb );
 printf("The complex dot product is:(%6.2f,%6.2f)\n",c.real,c.imag );
 return 0;
}
```

Calling BLAS Level 1 Function from C++

```
#include <complex>
#include <iostream>
#define MKL_Complex16 std::complex<double>
#include "mkl.h"
#define N 5
int main(){
  int n, inca = 1, incb = 1, i;
  std::complex<double> a[N], b[N], c;
  n = N;
  for( i = 0; i < n; i++ ){
     a[i] = std::complex<double>(i,i*2.0);
     b[i] = std::complex<double>(n-i,i*2.0);
 }
 zdotc(&c, &n, a, &inca, b, &incb );
 std::cout << "The complex dot product is: " << c << std::endl;
 return 0;
}
```


© https://software.intel.com/en-us/node/528729#IX_COMPLEX_BLAS_LEVEL_1_1

Using CBLAS Interface Instead of Calling BLAS Directly from C

```
#include <stdio.h>
#include "mkl.h"
typedef struct{ double re; double im; } complex16;
#define N 5
int main(){
  int n, inca = 1, incb = 1, i;
  complex16 a[N], b[N], c;
  n = N;
  for( i = 0; i < n; i++ ){
  a[i].re = (double)i; a[i].im = (double)i * 2.0;
  b[i].re = (double)(n - i); b[i].im = (double)i * 2.0;
}
cblas_zdotc_sub(n, a, inca, b, incb, &c );
printf( "The complex dot product is: ( %6.2f, %6.2f)\n", c.re, c.im );
return 0;
}
```

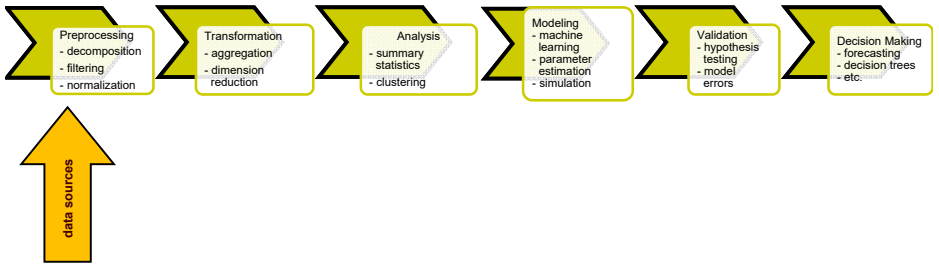

© https://software.intel.com/en-us/node/528729#IX_COMPLEX_BLAS_LEVEL_1_1

Makefile

```
CC=icpc
CFLAGS= -qopenmp -qopt-report -mkl

all: main

main:  main.cpp
     $(CC) -o main main.cpp $(CFLAGS)

clean:    rm main
```


Intel Data Analytics Library (DAAL)

- Open Source
- C++, Java, Python
- Cover all stages of data analytics



Intel Data Analytics Library (DAAL)

- Three main blocks

1) Data management
  - data sources
  - dictionaries
  - models
  - compression

Intel Data Analytics Library (DAAL)

- Three main blocks

2) Algorithms
  - analysis
  - training
  - prediction

Intel Data Analytics Library (DAAL)

- Three main blocks

3) Services
  - memory allocation
  - error handling
  - collections
  - shared pointers

Intel Data Analytics Library (DAAL)

- Processing
  - batch - send all and wait
  - online - incremental block processing
  - distributed - uses multiple nodes
- Heavily uses MKL and IPP

Intel Integrated Performance Primitives (IPP)

- Processing of
  - signal
  - images
  - video, and
  - data
- Windows, OS X, Linux, Android, VxWorks
- Interfaces with C

Intel Integrated Performance Primitives (IPP)

Description	Domain Code	*.h	Prefix
colors	CC	ippcc.h	ippi
string	CH	ippch.h	ipps
core func	CORE	ippcore.h	ipp
cryptography	CP	ippcp.h	ipps
computer vision	CV	ippcv.h	ippi
data compression	DC	ippcd.h	ipps
image processing	I	ippi.h	ippi
signal processing	S	ipps.h	ipps
vector math	VM	ippvm.h	ipps
embedded func	E	ippe.h	ipps

Intel Integrated Performance Primitives (IPP)

- Includes a huge amount of functions
- Optimized for OpenMP and AVX-512
- Can be used with any x86 compiler (!)
- Compatible with MS Visual Studio
- https://software.intel.com/en-us/ipp-dev-guide

Performance Libraries

- Can work directly in MCDRAM (fast)
- or in the main memory (slower)

Call:

```
mkl_set_memory_limit(MKL_MEM_MCDRAM, mbytes);
```

or set env variable:

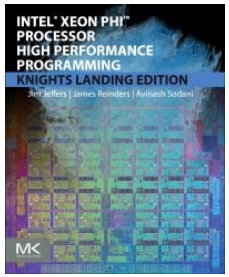
```
export MKL_FAST_MEMORY_LIMIT="1024"
```

use

```
mkl_malloc() etc.
```

Reading

- Intel Xeon Phi Processor High Performance Programming Knights Landing Edition
- James Jeffers, James Reinders, and Avinash Sodani
- ISBN: 9780128091944
- Morgan Kaufmann
- Chapter 13



Reading

- IPP
- https://software.intel.com/en-us/ipp-dev-guide