Lecture 8:

Caffe - CPU Optimization

boris. ginzburg@intel.com

Agenda

1. Profiling of application with Vtune
2. Caffe with BLAS
3. Parallelization with OpenMP

VTUNE: GETTING STARTED

Vtune: getting started

1. get non-commercial license and install Intel Parallel Studio XE 2013: https://software.intel.com/en-us/non-commercial-software-development (Includes Intel C++ Compiler (icc), Vtune, and MKL)
2. Buid application as usual (all optimization, but with debug information (-02 -g flag)
3. Run amplxe-gui
   - Create vtune project
   - Run basic Hot-spot analysis
   - Analyze performance

Exercise: Analyze caffe training using mnist example.

CAFFE WITH BLAS

Caffe with BLAS

Caffe is based on BLAS. CPU:

- OpenBLAS: http://www.openblas.net/ - very good open source library
- Intel MKL https://software.intel.com/en-us/intel-mkl - even better ☺, closed source , need license
- ATLAS http://math-atlas.sourceforge.net/ - slow

GPU:

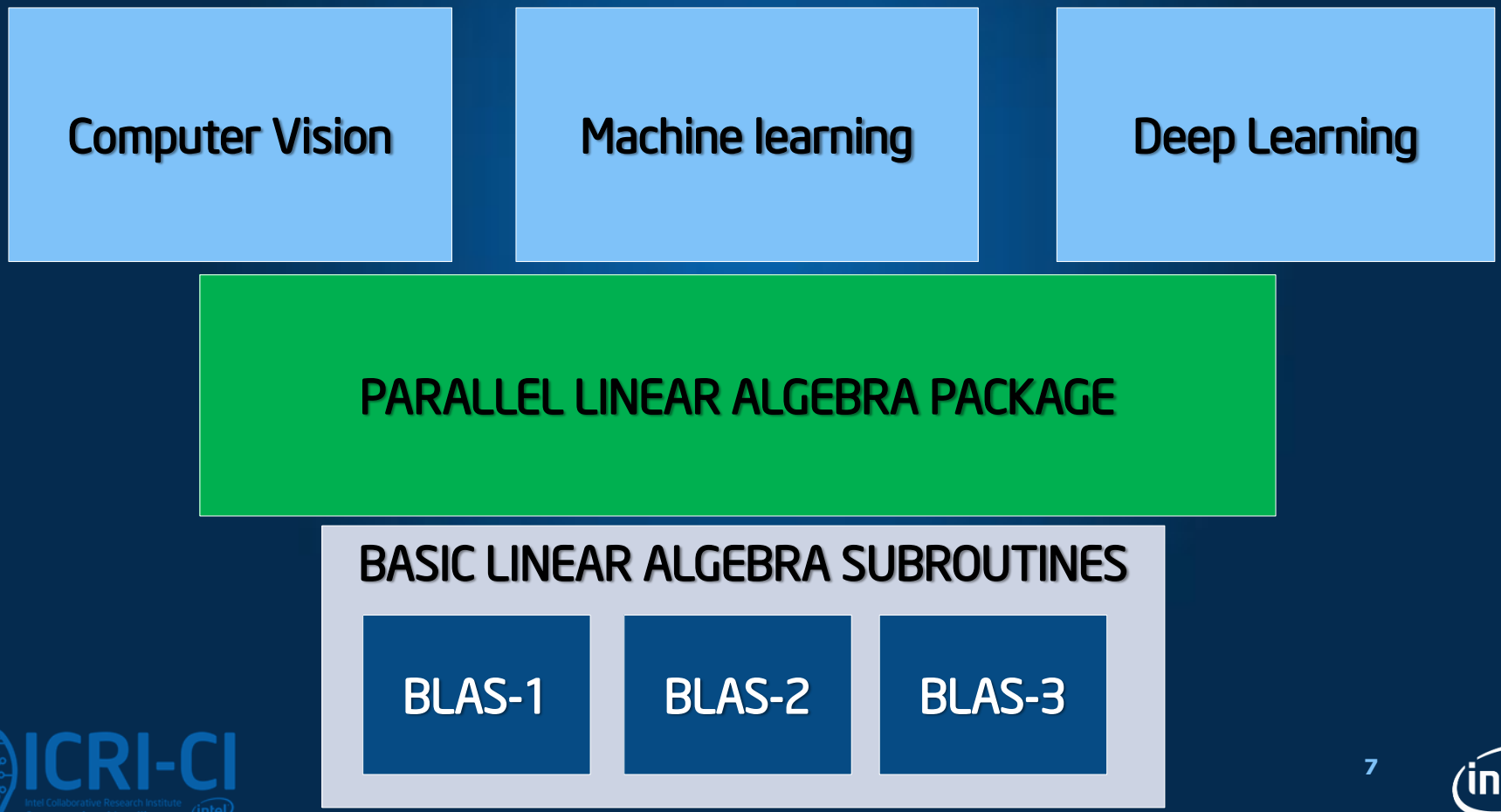
- cuBLAS (part of toolkit): https://developer.nvidia.com/cublas

Basic Linear Algebra Subroutines –set of low-level kernel subroutines for linear algebra:

1. BLAS1: vector – vector operations;
2. BLAS2: matrix – vector operations (e.g. "matrix vector multiply");
3. BLAS3: matrix – matrix operations (like matrix – matrix multiply).

BLAS: Foundation for Math Computing

BLAS is used as a building block in higher-level math libraries as LINPACK, MKL or PLASMA etc.



7

Exercise

1. Switch between ATLAS, OpenBLAS, and MKL in Makefile.config, compare performance on CIFAR-10
2. Download new version of OpenBLAS and build it. Compare performance.

OPENMP

Projects

OpenMP :

- an easy, portable and scalable way to parallelize applications for many cores.
- Multi-threaded, shared memory model (like pthreads)
- a standard API +
- omp pragmas are supported by major C/C++ , Fortran compilers (gcc, icc, etc).

A lot of good tutorials on-line:

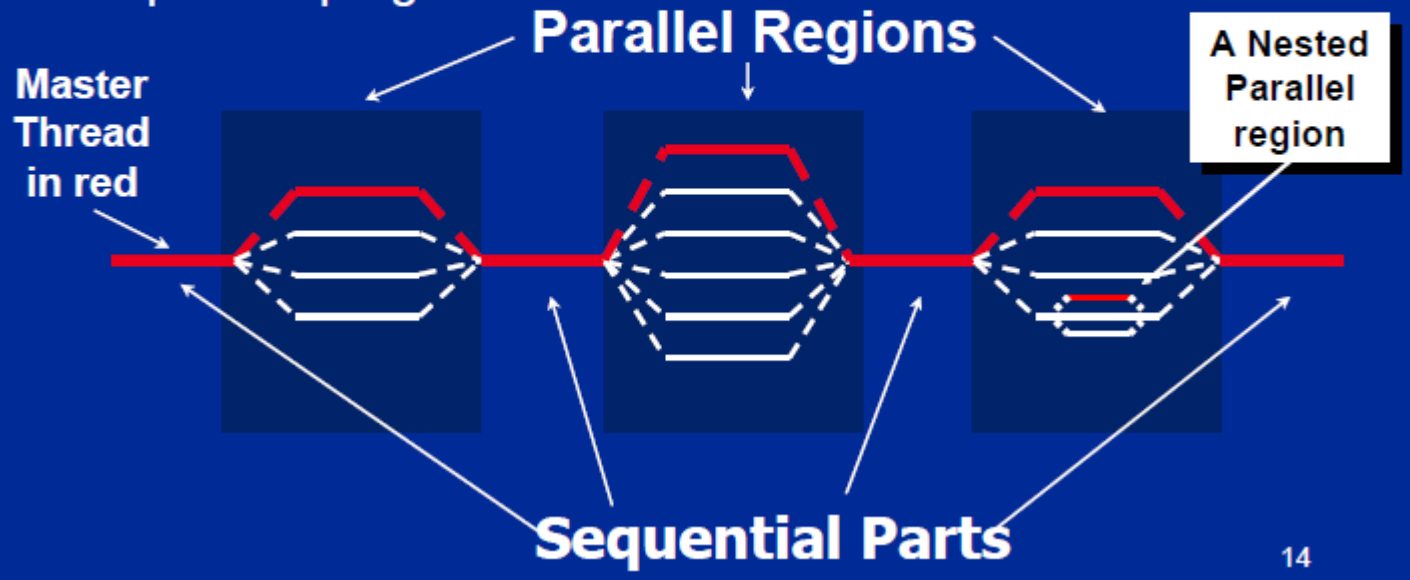
https://computing.llnl.gov/tutorials/openMP/

http://openmp.org/mp-documents/omp-hands-on-SC08.pdf

OpenMP programming model

Fork – Join parallelism

- Master thread spawns a team of threads as needed.
- Parallelism added incrementally until performance goals are met: i.e. the sequential program evolves into a parallel program.



Example 1

```
int main (int argc, char *argv[ ])  {
    int i;
    float a[N], b[N], c[N];

    for (i=0; i < N; i++) {
        a[i] = b[i] = 1.0;
    }

    for (i=0; i<N; i++) {
        c[ i ] = a[ i ] + b[ i ]
     }
}
```

Example 1

```
#include <omp.h>
int main (int argc, char *argv[ ])  {
    int i;
    float a[N], b[N], c[N];
#pragma omp parallel for
    for (i=0; i < N; i++) {
        a[i] = b[i] = 1.0;
    }
#pragma omp parallel for
    for (i=0; i<N; i++) {
        c[ i ] = a[ i ] + b[ i ]
     }
}
```

Example 1

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#define N   100
int main (int argc, char *argv[ ])  {
    int nthreads, tid, i;
    float a[N], b[N], c[N];
    nthreads = omp_get_num_threads();
    printf("Number of threads = %d\n", nthreads);
#pragma omp parallel for
    for (i=0; i < N; i++) {
        a[i] = b[i] = 1.0;
    }
```

Example 1

```
#pragma omp parallel for
   for (i=0; i<N; i++)    {
       c[ i ] = a[ i ] + b[ i ];
       tid = omp_get_thread_num();
       printf("Thread %d: c[%d]= %f\n", tid , i , c[ i ]);
    }
}
```

Compiling, linking etc

1. You need to add flag –fopenmp to gcc:
   gcc -fopenmp omp_vecadd.c -o vecadd
   icc -openmp omp_vecadd.c -o vecadd
2. Control number of threads through
   setenv OMP_NUM_THREADS 8

Exercise

1. Implement:
   - vector dot-product: c=<x,y>
   - matrix-matrix multiply,
   - 2D matrix convolution
2. Add openmp support to relu, and max-pooling layers