OpenACC
Lecture 1

Manfred Liebmann
Technische Universität München
Chair of Optimal Control
Center for Mathematical Sciences, M17
manfred.liebmann@tum.de

December 22, 2015

OpenACC Directives for Accelerators

The OpenACC Application Program Interface describes a collection of compiler directives to specify loops and regions of code in standard C, C++ and Fortran to be offloaded from a host CPU to an attached accelerator. OpenACC is designed for portability across operating systems, host CPUs, and a wide range of accelerators, including APUs, GPUs, and many-core coprocessors.

OpenACC Resources and Examples

`http://www.openacc.org`

OpenACC Programming Guide

`http://www.openacc.org/sites/default/files/OpenACC_Programming_Guide_0.pdf`

OpenACC Specification

The directives and programming model defined in the OpenACC API documents allow programmers to create high-level host+accelerator programs without the need to explicitly initialize the accelerator, manage data or program transfers between the host and accelerator, or initiate accelerator startup and shutdown.

OpenACC 2.5 Specification (November 2015)

`http://www.openacc.org/sites/default/files/OpenACC_2pt5.pdf`

OpenACC 2.5 Quick Reference Guide

`http://www.openacc.org/content/openacc-25-quick-reference-guide`

Complex Data Management Tech Report

`http://www.openacc.org/sites/default/files/TR-14-1.pdf`

OpenCC Toolkit

The OpenACC Toolkit from NVIDIA offers scientists and researchers a simple way to accelerated scientific computing without significant programming effort. Simply insert hints (or directives) in C or Fortran code and the OpenACC compiler runs the code on the GPU.

`https://developer.nvidia.com/openacc`

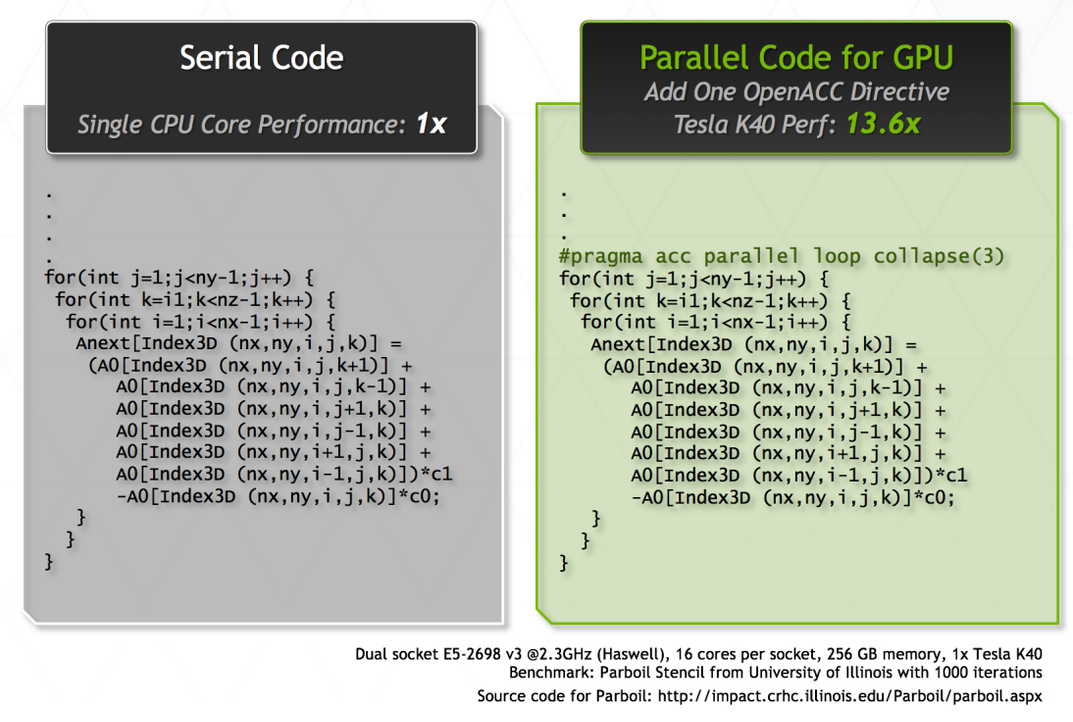

Figure 1: How OpenACC works

PGI Accelerator C/C++ Workstation Compiler Suite

To activate the PGI compiler suite on mephisto add the paths below to your `.bashrc` file.

```
export PATH=/share/apps/pgi/linux86-64/15.7/bin:$PATH
export MANPATH=$MANPATH:/share/apps/pgi/linux86-64/15.7/man
export LM_LICENSE_FILE=$LM_LICENSE_FILE:/share/apps/pgi/license.dat
```

Alternative: Download the *free* OpenACC Toolkit from Nvidia for your own machine!

`http://www.nvidia.com/object/openacc-toolkit.html`

OpenACC Compiler Configuration

OpenACC uses compiler directives, `#pragma acc ...`, to express parallel constructs for accelerators. OpenACC inherits many features and constructs form OpenMP.

```
#include<iostream>
using namespace std;

void vector_set(float *x, float a, int n)
{
    #pragma acc kernels
    for(int i = 0; i < n; i++) x[i] = a;
}

int main(int argc, char** argv)
{
    int n = 256 * 1024;
    float *z = new float[n]();
    vector_set(z, 2.0, n);

    bool good = true;
    for(int i = 0; i < n; i++) if (z[i] != 2.0) good = false;
    cout << "Check: " << (good ? "True" : "False") << ", z[0]: " << z[0] << endl;

    return 0;
}
```

Building and Executing OpenACC Programs

OpenACC requires a compatible C/C++ compiler to build the programs. The compiler flag `-acc` must be passed to the PGI compiler to enable the OpenACC features.

```
pgcpp -O3 -acc -o basic basic.cpp
```

OpenACC programs can be executed like standard programs.

```
./basic
```

OpenACC Tips and Tricks

Enable compiler info for OpenACC with the `-Minfo=accel` flag and generate debug info with the `-ta=tesla:lineinfo` flag.

```
pgcpp -O3 -ta=tesla:lineinfo -Minfo=accel -acc -o add add.cpp
```

Read the compiler info generated!

Use the Nvidia Visual Profiler `nvvp` to profile the OpenACC code.

```
nvvp
```

Restrict Your Pointers!

In computer programming, aliasing refers to the situation where the same memory location can be accessed using different names. Use the `restrict` keyword to explicitly tell the OpenACC compiler that no aliasing will occur.

```
void vector_add(float *restrict x, float *restrict y, float *restrict z, int n)
{
#pragma acc kernels copyin(x[0:n], y[0:n]) copyout(z[0:n])
    for(int i = 0; i < n; i++) z[i] = x[i] + y[i];
}
```

The OpenACC compiler can only generate an efficient kernel if no aliasing is assumed.

Gangs, Workers, Vectors

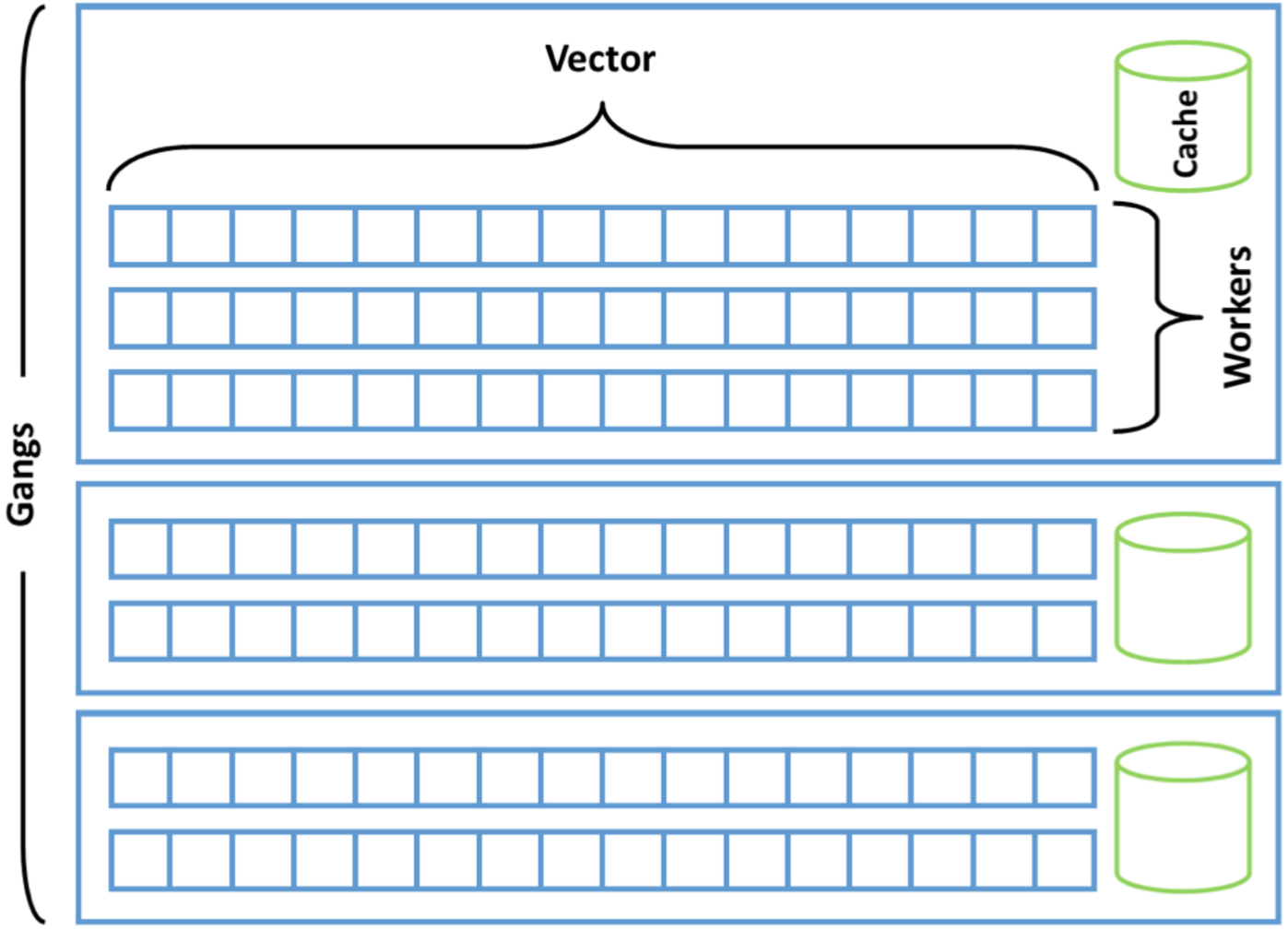

Figure 2: Gangs = Grids, Workers = Blocks, Vectors = Warps