OpenMP
Lecture 1

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

December 7, 2015

OpenMP Application Programming Interface

OpenMP (Open Multi-Processing) is an Application Program Interface (API), jointly defined by a group of major computer hardware and software vendors. OpenMP provides a portable, scalable model for developers of shared memory parallel applications. The API supports C/C++ and Fortran on a wide variety of architectures.

OpenMP Tutorial:

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

The OpenMP Architecture Review Board

The OpenMP ARB mission is to standardize directive-based multi-language high-level parallelism that is performant, productive and portable. Jointly defined by a group of major computer hardware and software vendors and major parallel computing user facilities, the OpenMP API is a portable, scalable model that gives shared-memory parallel programmers a simple and flexible interface for developing parallel applications on platforms ranging from embedded systems and accelerator devices to multicore systems and shared-memory systems.

`http://openmp.org`

OpenMP 4.5 Complete Specifications (November 2015)

`http://openmp.org/wp/openmp-specifications/`

Lecture focus: OpenMP 3.1 Complete Specifications & Summary Card C/C++

`http://www.openmp.org/mp-documents/OpenMP3.1.pdf`

`http://www.openmp.org/mp-documents/OpenMP3.1-CCard.pdf`

OpenMP API Components

OpenMP provides an application program interface (API) to explicitly express multi-threaded shared memory parallelism.

- OpenMP is comprised of three primary API components
  - Compiler directives
  - Runtime library routines
  - Environment variables

OpenMP Compiler Directives

OpenMP uses compiler directives, `#pragma omp ...`, to express parallel constructs. Furthermore the runtime library provides routines to control the execution environment of OpenMP. The program execution can also be influenced by special environment variables.

```
#include <iostream>
#include <omp.h>
using namespace std;

int main(int argc, char** argv)
{
    omp_set_num_threads(4);

    #pragma omp parallel
    {
        int id = omp_get_thread_num();
        int num = omp_get_num_threads();
        cout << "id: " << id << " threads: " << num << endl;
    }
}
```

Building and Executing OpenMP Programs

OpenMP requires a compatible C/C++ compiler to build the programs. The compiler flag `-fopenmp` must be passed to the GNU compiler to enable the OpenMP features.

```
g++ -O3 -fopenmp -o basic basic.cpp
```

OpenMP programs can be executed like standard programs. The program execution can be influenced by setting special environment variables.

```
export OMP_NUM_THREADS=2
./basic
```

OpenMP 3.1 API Implementations: Most modern compilers: gcc, icc, pgcc

Shared Memory Model

OpenMP is designed for multi-processor/core shared memory machines. The underlying hardware architecture can be shared memory with UMA or NUMA.

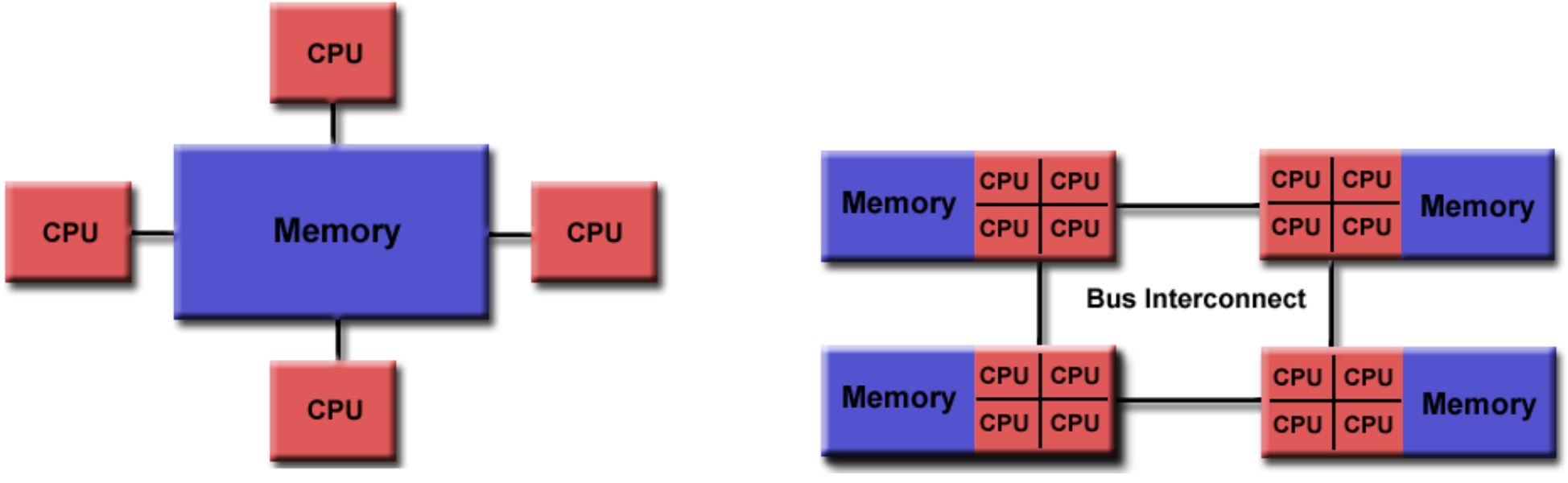

Figure 1: Uniform memory access (UMA) and nun-uniform memory access (NUMA) architectures