OpenMP

OPEN MULTI-PROCESSING

BY: BRIAN DIAZ

CSCI 330 SPRING 2021

OpenMP?

- OpenMP stands for Open Multi-processing
- OpenMP supports C, C++, and Fortran
- OpenMP was created in 1997
- OpenMP is not limited to any single OS, it can run on Linux, Windows, and Mac OS.

Open**MP** what even is this?

- **OpenMP uses API (Application program interface) that may be used to explicitly direct multi-threaded, shared memory parallelism.**
- **OpenMP contains 3 primary API components**
- **Compiler Directives**
- **Runtime Library Routines**
- **Environment Variables**

OpenMP is not..

- Designed to handle parallel I/O
- Required to check for data dependencies and data conflicts
- Guaranteed to make the most efficient use of shared memory
- Meant for memory parallel systems by itself

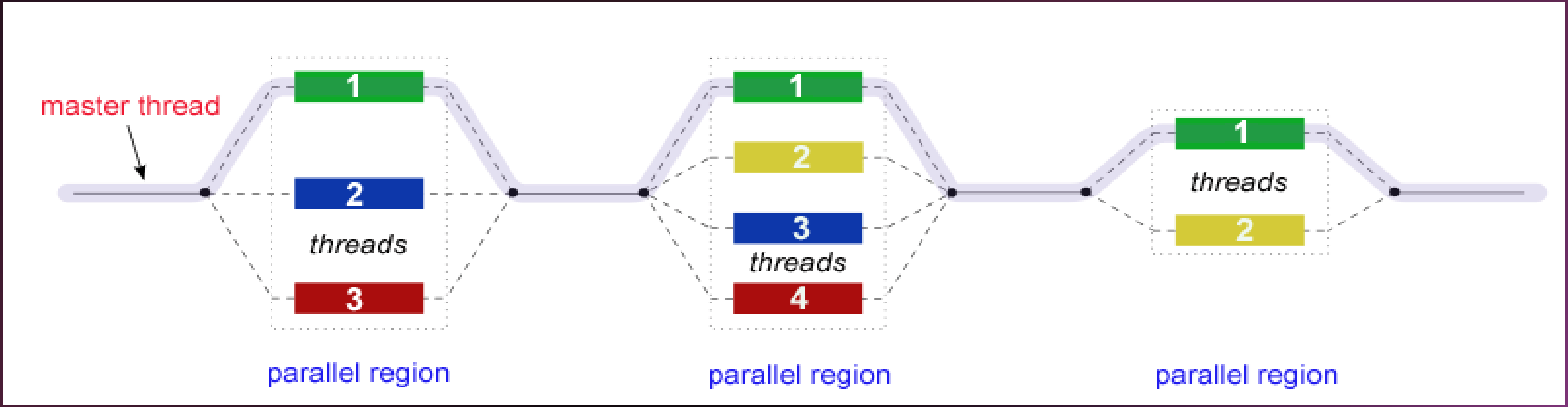
How Does OpenMP work?

Thread Based/Explicit Parallelism

- OpenMP programs accomplish parallelism using threads.
- A thread exist within the resources of a single process. Without the process they do not exist.
- Usually, the number of threads match the number of processors/cores however the program can contain more threads if wanted.
- OpenMP offers developers full control over parallelization.
- Parallelization can be simple or very complex, however it all depends on what you plan to do.

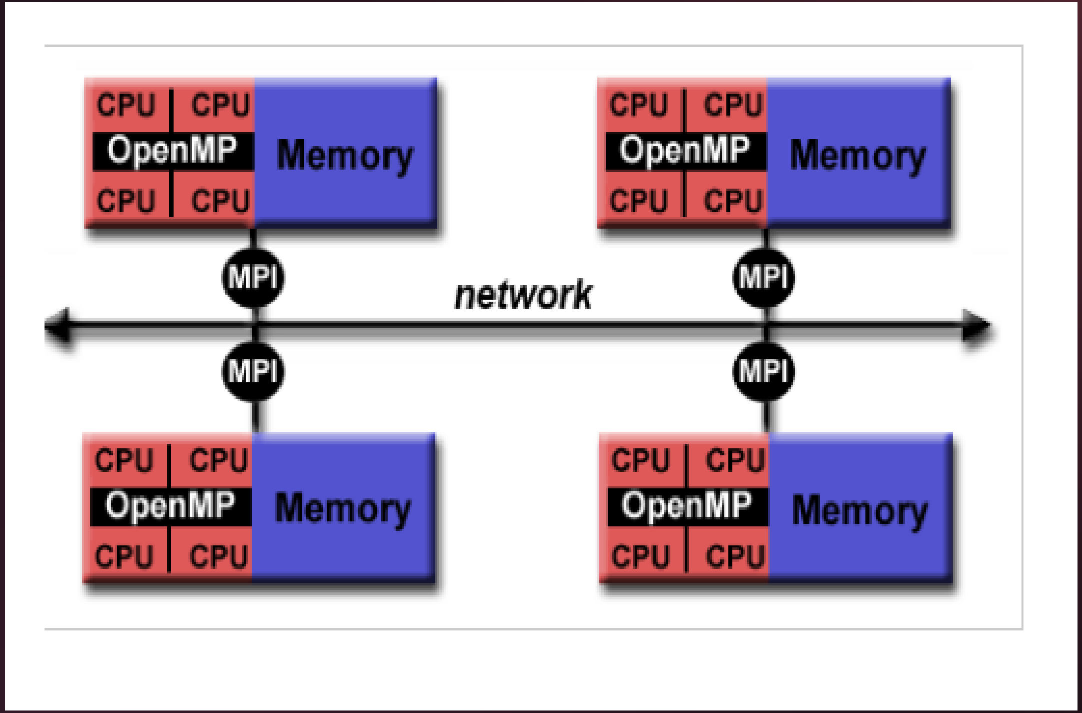
Fork-Join Model for OpenMP

- OpenMP uses a master thread and executes sequentially until the first parallel region is encountered.
- The master thread then creates a team of parallel threads.
- When the team thread is finished the region is synchronized and terminated leaving the master thread.
- The number of threads and regions that are compromised are arbitrary.



How Does OpenMP help HPC?

High Performance Computing(HPC)



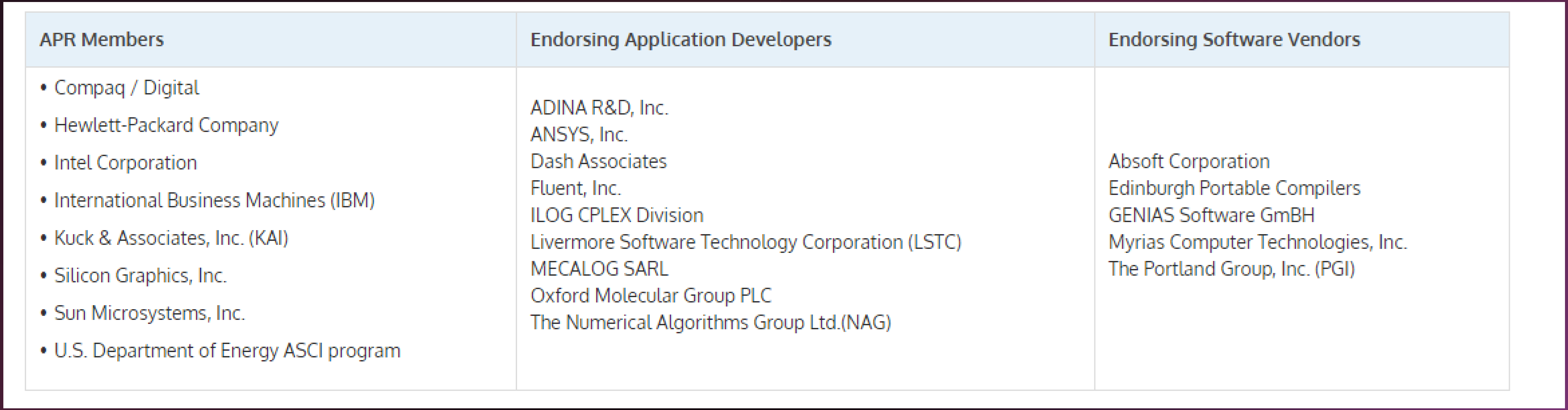
- **OpenMP helps HPC in having all the cores share access to main memory and provide less overhead.**
- **OpenMP and MPI combine for the distributed memory parallelism to function in HPC.**
- **This allows parallelism to be done to the full scale of a cluster**
- **MPI is required since OpenMP parallelism is limited to a single node.**

OpenMP Relates to MPI and OpenCL

- MPI works in complement to OpenMP
- OpenCL is like OpenMP however OpenMP has its key pros and cons
- The pros being it has better performance and can outperform OpenCL during compilation and during runtime.
- The cons being that OpenCL can provide more detailed expression of parallelism and it is way less work than OpenMP to incorporate.
- Now what software is useful over the other? OpenMP is easier to program than MPI however it can be used in complement to each other. OpenMP is also a better performer, and the code is easy to maintain and read. OpenMP seems like the option to choose over MPI however it all depends on what you try to do.

Who uses OpenMP?

- OpenMp is very popular among many different companies. Some of those found being Altair OptiStruct and GenASiS.
- Researching more on this matter I found this table for OpenMP



APR Members	Endorsing Application Developers	Endorsing Software Vendors
• Compaq / Digital • Hewlett-Packard Company • Intel Corporation • International Business Machines (IBM) • Kuck & Associates, Inc. (KAI) • Silicon Graphics, Inc. • Sun Microsystems, Inc. • U.S. Department of Energy ASCI program	ADINA R&D, Inc. ANSYS, Inc. Dash Associates Fluent, Inc. ILOG CPLEX Division Livermore Software Technology Corporation (LSTC) MECALOG SARL Oxford Molecular Group PLC The Numerical Algorithms Group Ltd.(NAG)	Absoft Corporation Edinburgh Portable Compilers GENIAS Software GmBH Myrias Computer Technologies, Inc. The Portland Group, Inc. (PGI)

What do the API components do for Open**MP**?

OpenMP Compiler Directives

- **Compiler Directives appear as comments and are ignored by the compiler. OpenMP Compiler Directives can be used for different reasons such as:**

1. **Spawning a parallel region**
2. **Dividing blocks of code among threads**
3. **Distributing loop iterations between threads**
4. **Serializing sections of code**

OpenMP Runtime Library Routines

- Runtime Library Routines in OpenMP are almost uncountable. The reasons you would use these routines would be for:

1. Querying a thread's unique identifier (thread ID), a thread's ancestor's identifier, the thread team size
2. Setting and querying nested parallelism
3. Setting, initializing and terminating locks and nested locks
4. Querying wall clock time and resolution

OpenMP Environment Variables

- **Environment Variables in OpenMP provide controlling of the execution of parallel code at run-time. These could be used to control stuff such as:**

1. **Setting the number of threads**
2. **Specifying how loop iterations are divided**
3. **Binding threads to processors**
4. **Setting thread stack size**

Works Cited

- "OpenMP FAQ" OpenMP. https://www.openmp.org/about/openmp-faq/
- "OpenMP in a nutshell." Bowdoin.edu. http://www.bowdoin.edu/~ltoma/teaching/cs3225-GIS/fall16/Lectures/openmp.html
- "OpenMP tutorial" Lawrence Livermore National Lab. https://hpc.llnl.gov/openmp-tutorial