2. Parallel Programming Languages

[1]

[1] M. Ganesh, MATH440/540, SPRING 2018

- [2]**The main aim of parallel computing is to partition a computation into subcomputations that can be executed in parallel**
- **Several actions must occur in parallel programs, independently of how they are described**
- **The actions include:**
  - ⋆ **Data distribution among processes**
  - ⋆ **Work distribution among processes**
  - ⋆ **Data exchange among processes**
  - ⋆ **Synchronization of different processes**
- **Some of the identified distinct paths for the development of applications software for parallel computers include:**

[2]M. Ganesh, MATH440/540, SPRING 2018

1. [3]**Extend an existing compiler to translate sequential programs into parallel programs**
2. **Extend an existing language with new operations that allow users to express parallelism**
3. **Add a new parallel language on top of an existing sequential language**
4. **Define a parallel language and compiler system**

- **The first three approaches have been tried by a few companies and resulted in some products such as:**
  - ⋆ **Parallel Software Products (company)**
  - ⋆ **CODE (Computationally Oriented Display Environment)**
  - ⋆ **Hence (Heterogeneous Network Computing Environment)**

[3]M. Ganesh, MATH440/540, SPRING 2018

- [4]**The main concern of the first three approaches is that the main focus is either on the language or on the compiler.**
- **Separate development of these two is likely to lead to difficulty in debugging codes with substantial complexity in application**
- **The fourth approach to create a parallel language and associated compiler systems gives the programmer the ability to express parallel operations explicitly**
- **One way for the fourth approach is to develop a parallel language from scratch.**
- **The programming language *OCCAM* is a well known example of this approach, which has a syntax strikingly different from traditional languages such as C and Fortran**
- **Another way in the fourth approach to support explicit parallelism is to add parallel constructs to an existing language**
- **C^* and Unified Parallel C (extensions of C) , and Fortran 90+ (variants of Fortran such as Fortran 95/2003 and HPF - High Performance Fortran) are examples of the fourth approach with extensions**

[4]M. Ganesh, MATH440/540, SPRING 2018

- [5]**For example, one of the key extension of C in C^* is the concept called *shape*:**
- **A *shape* specifies the way in which parallel data are organized**
- **A *shape* is an array like object, but while array elements must be manipulated one at a time, elements of a shape may be manipulated simultaneously**
- **For example, to set up a template for parallel data/matrix of shape/size 100×200, one may use the declaration**
- `shape [100][200]matrix;`
- **Then to create parallel data/matrix A, B, C, of size 100×200, use**
- `float:matrix A, B, C;`
- **The statements that C is a sum of A and B, in C^*, is**
- `with (matrix) { C = A+B; }`
- **Thinking Machines Corporation developed C^***

[5]M. Ganesh, MATH440/540, SPRING 2018

- [6]**UPC (Unified Parallel C) is a relatively recent extension of the C programming language, designed for high performance computing on cluster parallel machines**
- **Commercial UPC compilers are developed, for example, by HP and IBM, see** `http://h30097.www3.hp.com/upc/` **and** `http://www.alphaworks.ibm.com/tech/upccompiler`
- **Open source UPC compilers are available, for example, GCC UPC, see** `http://www.intrepid.com/upc.html`
- **UPC is a SPMD (Single Program Multiple Data) extension of C**
- **The SPMD is a single program parallel approach that involves only one program which will execute differently on different processors**
- **The differences can occur through either implicit mechanism such as compiler directives or explicit constructs such as references to the processor number executing the code**
- **See the above websites for details of UPC facilitating explicit parallelism (such as** `MYTHREAD`**) with a shared address space (such as** `shared`**)**

[6]M. Ganesh, MATH440/540, SPRING 2018

- [7]**Substantial extensions from Fortran 77 in Fortran 90+ (Fortran 95/2003, HPF) within the fourth approach facilitate parallelism**
- **For example, in Fortran 77 (or C), we need a double nested `DO` (or `for`) loops to add and/or subtract two $N \times M$ matrices A and B to get data C and/or D**
- **In Fortran 90+ this can be accomplished using `FORALL` statements**
- **The main difference between `DO` and `FORALL` in Fortran is that in the `DO` loop structure, execution must be in a strict order, while the statements in the `FORALL` construct may be executed in any order**
- **In particular, `FORALL` construct process the same set of elements in any order selected by the processor**
- **Fortran 95/2003 is an SPMD extension of Fortran 77/90**
- **HPF (High Performance Fortran) is a data parallel extension of Fortran 77/90**
- **The main focus of data parallelism is on distributing the data across different parallel computing nodes**

[7]M. Ganesh, MATH440/540, SPRING 2018

[8]**Most parallel programs are written using either**

- **MPI (Message Passing Interface) with a SPMD model:**
  - ⋆ **Usually for scientific applications with C or Fortran (with wrappers used to call C from Fortran and vice-versa)**
  - ⋆ **Scales easily: user controlled data layout**
  - ⋆ **Difficulty: send/receive matching, message packing/unpacking**

- **or using shared memory programming model with OpenMP**
  - ⋆ **Usually for non-scientific applications**
  - ⋆ **Easier to program: direct reads and writes to shared data**
  - ⋆ **Hard to scale: mostly limited to SMPs; no concept of locality**

[8]M. Ganesh, MATH440/540, SPRING 2018

- **[9]Next we consider basic techniques in Fortran 95/2003,**

- **Then, we will develop**
  parallel programming with MPI in F90+, C, and C++

- **This will then by followed by**
  parallel programming with OpenMP in F90+, C and C++

- **Subsequently, we will look into**
  MPMD models with codes written using F90, C, C++, and Python

- **Time permitting, we will also explore**
  parallel programming for GPU processors

[9]M. Ganesh, MATH440/540, SPRING 2018