Çankaya University

Department of Computer Engineering

Ceng 471 Parallel Computing

<u>Project :</u> **Parallel Programming Languages**

Özerk Önder 200311205

INTRODUCTION

- In the past 20 years, parallel computation has helped to solve many significant problems: especially those not implementable on sequential computers.
- Parallel computers represent an opportunity.

- This opportunity is driven by parallel (concurrent) programming languages that make high-performance machines usable and useful.

- Parallel languages allows users to design parallel algorithms as a set of concurrent actions mapped onto different computing elements.

- Cooperation among actions can be performed in several ways according to the selected paradigm.

- High-level languages might decrease both the design time and the execution time > Easier Approach to Parallelism for New Users.

- Typical issues in parallel programming are
    - process creation,
    - synchronization,
    - communication handling,
    - deadlock, and
    - process termination.

- These issues arise because that are many flows of control through the program (one per process).

- Languages should make the programming of multicomputers to be not much harder that programming sequential computers.

Shared Memory Paradigms

- The concept of shared memory is a useful way to separate program control flow issues from issues of data mapping, communication, and synchronization.

- Processes cooperate through a shared memory space where shared variables are stored.

- Some languages for parallel programming provide basic mechanisms for data sharing.

- **Shared Memory Languages:**
    - Linda,
    - Orca,
    - SDL,
    - OpenMP,
    - Pthreads,
    - Ease,
    - Opus,
    - Java.

Linda

- Linda provides an associative memory abstraction called tuple space.

- Threads communicate with each other only by placing tuples in and removing tuples from this shared associative memory.

- Sequential languages can be augmented with tuple space operations to create a new parallel programming language.

- Linda is called a coordination language because the tuple space abstraction coordinates, but is orthogonal to, the computation activities.

Orca

- Orca is a language based on a useful set of primitives for sharing of data among processes.

- The Orca system is a hierarchically structured set of abstractions.
    - At the lowest level, reliable broadcast is the basic primitive so that writes to a replicated structure can rapidly take effect throughout a system.
    - At the next level of abstraction, shared data are encapsulated in passive objects that are replicated throughout the system.

- On these levels, Orca itself provides an object-based language to create and manage objects.

OpenMP

- OpenMP is a library (application program interface - API) that supports parallel programming on shared memory parallel computers.

- OpenMP has been developed by a consortium of vendors of parallel computers (DEC, HP, Sun, Intel, …) with the aim to have a standard programming interface for parallel shared-memory machines.

- The OpenMP functions can be used inside Fortran, C and C++ programs.

- They allow the parallel execution of code, the definition of shared data and synchronization of processes.

Java

- An important shared-memory programming language is Java that is popular because of its connection with platform-independent software delivery on the Web.

- Java is an object-oriented language that supports the implementation of concurrent programs by process (called threads) creation and execution.
- To use Java on distributed-memory parallel computer there are different solutions:
  - sockets,
  - RMI (Remote Method Invocation),
  - Java + CORBA.

PVM (Parallel Virtual Machine)

- PVM (Parallel Virtual Machine) is a toolkit currently used to implement parallel applications on heterogeneous computers.

- The PVM environment provides primitives for process creation and message passing that can be incorporated into existing procedural languages.

- PVM runs on many platforms from several vendors. In a PVM program a process can run on a workstation and another process can run on a supercomputer.

- For these reasons PVM is widely used and programs are portable,

  BUT

- It offer a low-level programming model. Using PVM, programmers must do all of the decomposition, placement, and communication explicitly.

HPF (High Performance Fortran)

- HPF is a language for programming computationally intensive scientific applications on SIMD, MIMD and vector processors.

- HPF is based on exploitation of loop parallellism.

- Iterations of the loop body that are conceptually independent can be executed concurrently.

C*

- The data-parallel C* is an extension of C language.

- C* was designed by Thinking Machines Corp. to program the Connection Machine.

- However, C* can be used to program several multicomputers using the data parallel approach.

- In this way, each processing element executes, in parallel, the same statement for each instance of the specified data type.

MPL

- MPL (Mentat Programming Language) is a parallel extension of C++ that combines
  - the object-oriented model with
  - the data-driven computation model.

- Data-driven model: parallel operations are executed on independent data when they are available.

- The data-driven model supports high degree of parallelism, while the object-oriented paradigm hides much of the parallel environment from a user.

- MPL implements both inter-object parallelism (one process per object) and intra-object parallelism (more processes per object).

- The compiler generates code to build and execute data dependency graphs. Thus parallelism in MPL is largely transparent to the programmer.

HPC++

- High Performance C++ is a standard library for parallel programming based on the C++ language.

- HPC++ is composed of two levels:
  - Level 1- consists of a specification for a set of class libraries based on the C++ language.
  - Level 2 -provides the basic language extensions and runtime library needed to implement the full HPC++.

- There are two conventional modes of executing an HPC++ program.

- The first is multi-threaded shared memory where the program runs within one context.
  - Parallelism comes from the parallel loops and the dynamic creation of threads.
  - This model of programming is very well suited to modest levels of parallelism.

- The second mode of program execution is an explicit SPMD model where n copies of the same program are run on n different contexts.
  - Parallelism comes from parallel execution of different tasks.
  - This model is well suited for massively parallel computers.

Conclusion

- A parallel programming language should

  - be easy to program, by providing mechanisms for
    - <u>Decomposition</u> of a program into parallel threads;
    - <u>Mapping threads</u> to processors;

        - Communication and synchronization among threads.
    - provide a software development methodology;
    - be architecture-independent;
    - have guaranteed performance on different architectures;
    - provide cost measures of programs.

- Parallel programming languages support the implementation of high-performance applications in many areas: from the Internet to computational science.

- New models, methods and languages allow users to develop more complex programs with minor efforts.