--- What is Mentat? ---

Mentat is an object-oriented parallel processing system designed to directly address the difficulty of developing architecture-independent parallel programs. The fundamental objectives of Mentat are to (1) provide easy-to-use parallelism, (2) achieve high performance via parallel execution, and (3) facilitate the execution of applications across a wide range of platforms. The Mentat approach exploits the object-oriented paradigm to provide high-level abstractions that mask the complex aspects of parallel programming, including communication, synchronization, and scheduling, from the programmer. Instead of managing these details, the programmer concentrates on the application. The programmer uses application domain knowledge to specify those object classes that are of sufficient computational complexity to warrant parallel execution.

Mentat combines a medium-grain, data-driven computation model with the object-oriented programming paradigm and provides automatic detection and management of data dependencies. The data-driven computation model supports high degrees of parallelism and a simple decentralized control, while the use of the object-oriented paradigm permits the hiding of much of the parallel environment from the programmer. Because Mentat uses a data-driven computation model, it is particularly well-suited for message passing, non-shared memory architectures.

There are two primary aspects of Mentat: the Mentat Programming Language (MPL) and the Mentat run-time system. The MPL is an object-oriented programming language based on C++ that masks the difficulty of the parallel environment from the programmer. The granule of computation is the Mentat class member function. The programmer is responsible for identifying those object classes whose member functions are of sufficient computational complexity to allow efficient parallel execution. Instances of Mentat classes are used exactly like C++ classes, freeing the programmer to concentrate on the algorithm, not on managing the environment. The data and control dependencies between Mentat class instances involved in invocation, communication, and synchronization are automatically detected and managed by the compiler and run-time system without further programmer intervention. By splitting the responsibility between the compiler and the programmer, we exploit the strengths and avoid the weaknesses of each. The underlying assumption is that the programmer can make better decisions regarding granularity and partitioning, while the compiler can better manage synchronization. This simplifies the task of writing parallel programs, making parallel architectures more accessible to non-computer scientist researchers.

The run-time system supports parallel object-oriented computing on top of a data-driven, message-passing model. It supports more than just method invocation by remote procedure call (RPC). Instead the run-time system supports a graph-based, data-driven computation model in which the invoker of an object member function need not wait for the result of the computation, or for that matter, ever receive a copy of the result. The run-time system constructs program graphs and manages communication and synchronization. Furthermore, the run-time system is portable across a wide variety of MIMD architectures and runs on top of the existing host operating system. The underlying operating system must provide process support and some form of inter-process communication.

Mentat has been distributed to over 100 sites in the US and abroad, and has been successfully applied to a wide-variety of real-world problems. We invite new users who wish to learn more about Mentat to write to us for documentation at mentat@Virginia.EDU

or to download the system and documentation directly, see the section on Distribution.

The Mentat system is available FREE. The systems supported in this release are the Sun 4 w/ SunOS 4.x, Sun 4 w/ SunOS 5.x (Solaris), Silicon Graphics, HP 9000/700, IBM RS6000, and Linux.

For more information see the section containing the Mentat Documentation and Manuals.

Last Modified : 22 Jan 1996. Mark Hyett (mrh2e@virginia.edu)