Parallel Programming with CHARM

CHARM is a machine independent parallel programming system. Programs written using this system will run unchanged on MIMD machines with or without a shared memory. It provides high-level mechanisms and strategies to facilitate the task of developing even highly complex parallel applications.

Charm programs are written in C with a few syntactic extensions. It is possible to interface to other languages such as FORTRAN using the foreign language interface that C provides. Charm++ is the C++-based parallel object oriented language having all features of Charm, which supports multiple inheritance, late bindings, and polymorphism.

Platforms: The system currently runs on Intel's iPSC/860, iPSC/2 and Paragon, Thinking Machines CM-5, nCUBE/2, IBM SP-2, Encore Multimax, Sequent Symmetry, single-processor UNIX machines, and networks of UNIX workstations. We plan to port it to the KSR-1, Cray T3D, Convex Exemplar and other parallel machines as they become available.

The design of the system is based on the following tenets:

1. **Efficient Portability:** Portability is an essential catalyst for the development of reusable parallel software. Charm/Charm++ programs run unchanged on MIMD machines with or without a shared memory. The programming model induces better data locality, allowing it to support machine independence without losing efficiency.

2. **Latency Tolerance:** Latency of communication — the idea that remote data will take longer to access — is a significant issue common across most MIMD platforms. *Message-driven execution*, supported in CHARM , is a very useful mechanism for tolerating or hiding this latency. In message driven execution (which is distinct from just message-passing), a processor is allocated to a process only when a message for the process is received. This means when a process blocks, waiting for a message, another process may execute on the processor. It also means that a single process may block for any number of distinct messages, and will be awakened when any of these messages arrive. Thus, it forms an effective way of scheduling a processor in the presence of potentially large latencies.

3. **Dynamic Load Balancing:** Dynamic creation of work is necessary in many applications. CHARM supports this by providing dynamic (as well as static) load balancing strategies.

4. **Specific Information Sharing Modes:** A major activity in a parallel computation is creation and sharing of information. Information is shared in many specific modes. The system provides six information sharing modes, each of which may be implemented differently and efficiently on different machines.

5. **Reuse and Modularity:** It should be possible to develop parallel software by reusing existing parallel software. CHARM supports this with a well-developed "module" construct and associated mechanisms. These mechanisms allow for compositionality of modules without sacrificing the latency-tolerance. With them, two modules, each spread over hundreds of processors, may exchange data in a distributed fashion.

The Programming Model: Programs consist of potentially medium-grained processes (called chares), and a special type of replicated process. These processes interact with each other via messages and any of the other information-sharing abstractions. There may be thousands of medium-grained processes on each processor, or just a few, depending on the application. The "replicated processes" can also be used for implementing novel information sharing abstractions, distributed data structures, and intermodule interfaces. The system can be considered a concurrent object-oriented system with a clear separation between sequential and parallel objects.

Reusable Libraries: The modularity-related features make the system very attractive for building library modules that are highly reusable because they can be used in a variety of data-distributions. We have just begun the process of building such libraries, and have a small collection of library modules. However, we expect such libraries, contributed by us and other users, to be one of the most significant aspects of the system.

Regular and Irregular Computations: For regular computations, the system is useful because it provides portability, static load balancing, and latency tolerance via message driven execution, and facilitates construction and flexible reuse of libraries. The system is unique for the extensive support it provides for highly irregular computations. This includes management of many medium-grained processes, support for prioritization, dynamic load balancing strategies, handling of dynamic data-structures such as lists and graphs, etc. The specific information sharing modes are especially useful for such computations.

Distribution: The system is available (for research use and evaluation) by anonymous ftp from a.cs.uiuc.edu (128.174.252.1), under the directory pub/CHARM or from the WWW at http://charm.cs.uiuc.edu . The files there include a manual, installation scripts, a README file that explains how to install the system on your machine, the requisite source/object files, and papers and reports from the Parallel Programming Laboratory.

Associated Tools:

DagTool allows specification of dependences between messages and sub-computations within a single process, provides a pictorial view of this dependence graph, and simplifies management of message-driven execution.

Projections is a performance visualization and feedback tool. Projections has a much more refined understanding of user computation than is possible in traditional tools because it is language-specific. Thus it can provide much specific feedback about entities in the user program such as objects and messages. It also incorporates a unique expert performance analysis component which can provide recommendations for improving performance.

Future Plans: Some of projects based on CHARM include:

DP-Charm A Data Parallel Language, providing a subset of HPF features.

Other base languages The basic ideas and techniques used in CHARM are independent of the base language used. We plan to incorporate them in other languages including FORTRAN and Scheme.

Debugging Tools Based on the specificity of information available to the runtime system, we plan to develop both trace-based and run-time debugging tools for CHARM .

Contact: L.V. Kale kale@cs.uiuc.edu
Department of Computer Science (217) 244-0094
University of Illinois at Urbana Champaign
1304 W. Springfield Ave., Urbana, IL-61801