

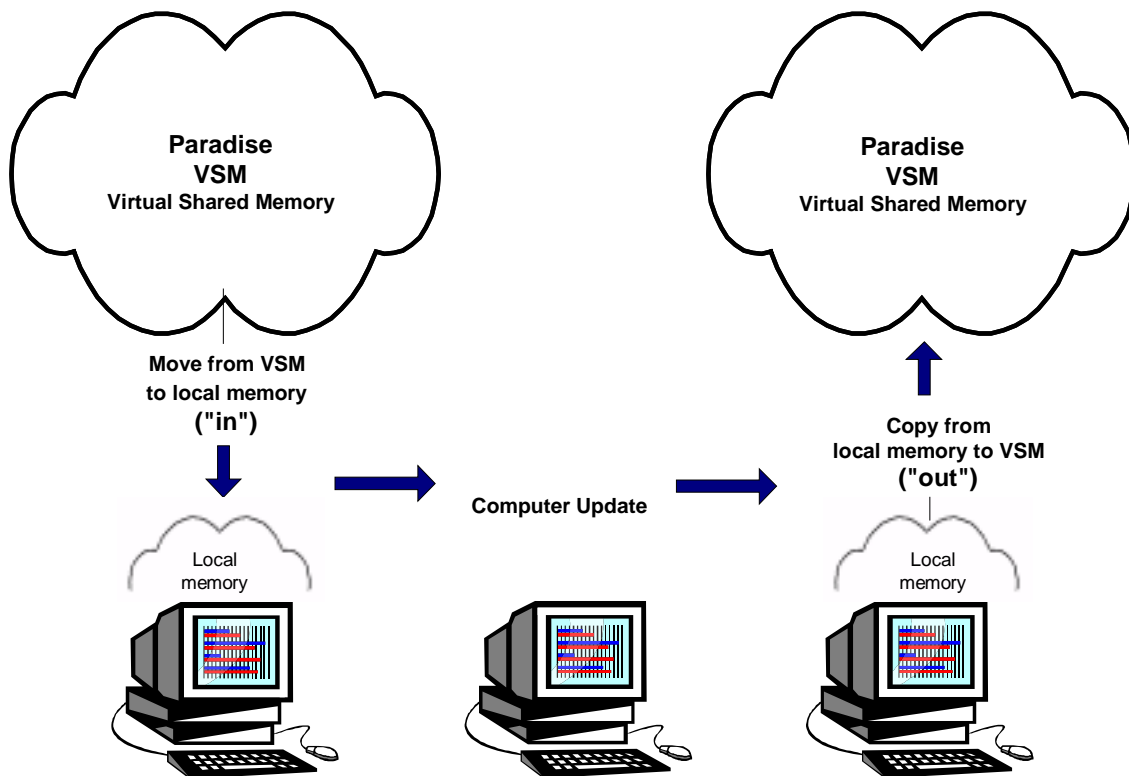
Paradise for Programmers Using Message Passing

Paradise® Virtual Shared Memory (VSM) middleware from Scientific Computing Associates Inc. can be used in conjunction with message passing systems, such as MPI and PVM, to enhance communication capabilities. Virtual Shared Memory is an advanced form of Distributed Shared Memory.

Many codes involve at least some data that, logically, should be equally accessible to all processes at all times. Examples include running maxima and minima, index generators, and shared resource semaphores.

Paradise permits developers to combine shared memory and message passing styles of parallel programming in the same program. Data that should be equally accessible to all processes is shared via Paradise's distributed shared memory. This minimizes the need for message passing and improves overall performance of the code.

As illustrated below, the VSM on a Paradise server allows data to be shared by participating processes which may be implemented in different programming languages running on different processors.



For example, if a process wants to increment a shared counter, it can use the Paradise idiom:

```
in@ts("count", ?val);  
val = val + 1  
out@ts("count", val);
```

This is as opposed to the several pages of code required in MPI to implement the same shared counter.

Conceptually, such shared data, should have an existence distinct from any particular process. Message passing is a mechanism that enables data to move between processes, but provides no data storage independent of these processes.

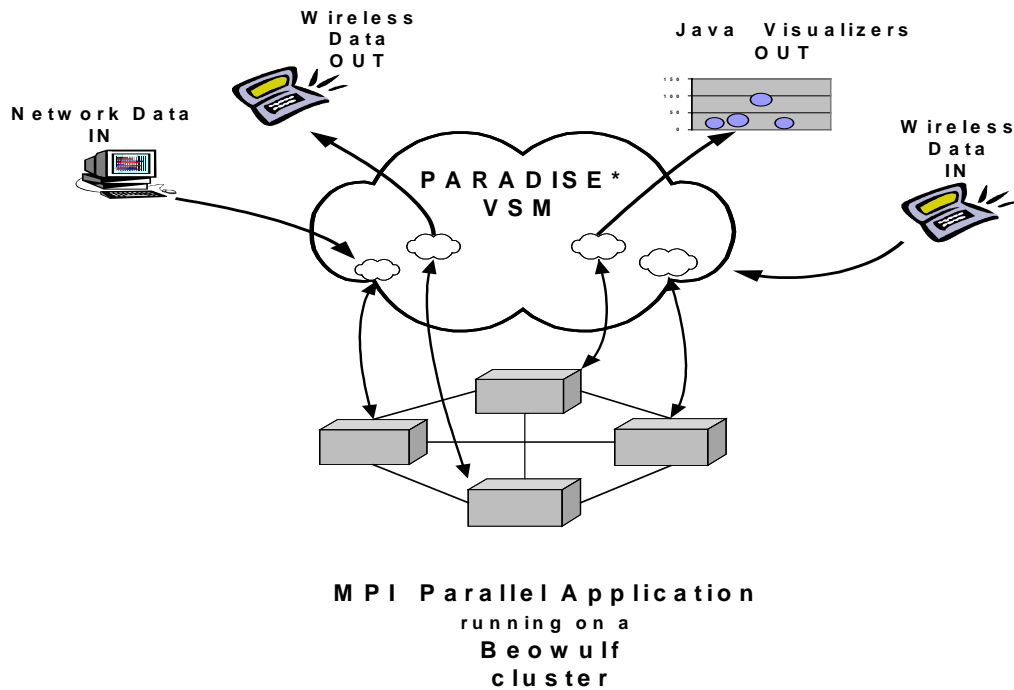
The Paradise VSM provides a global, content-addressable memory, which is a natural home for such shared data. A developer uses a Paradise VSM with a few simple commands, while the low-level details of interprocessor communication and VSM management are taken care of automatically. This makes Paradise easy to learn and to use. This can greatly reduce the complexity of creating, debugging, and maintaining parallel applications while still providing high performance. Investments in programmer training and application development are fully protected.

Message passing systems do not scale well, conceptually or pragmatically, to distributed applications and ensembles of applications. Yet few applications exist in a vacuum.

Utilizing Paradise, developers can overcome this limitation and create “ensemble applications” from separate independent programs that can even run on different machines. For example, in the illustration below, computers in a Beowulf cluster running a parallel application implemented with MPI can put results into a Paradise VSM as they are computed or on a periodic basis. A separate visualization application (for example, written in Java) can retrieve the data from the Paradise VSM and display it graphically.

Applications could run concurrently, in which case the visualization application would provide a live snapshot of the data that could be used for “computational steering.” But they could also run asynchronously at completely different times, with the visualization application displaying, for example, summary information about the results from the previous day.

.....



Paradise is multilingual and multiplatform, enabling Message Passing/Paradise applications to "plug into" a larger ensemble of Paradise-enabled applications. Even multiple parallel programs can be coordinated to work together. Paradise has APIs to support the C, C++, Java, Python, and Fortran programming languages. Paradise runs on Linux, Windows, and most UNIX based computers.

Using Paradise, applications can share data asynchronously and anonymously without the identity of all participating processes being known a priori. Programs sharing data need not run concurrently, and applications can detach from and reattach to Paradise VSMs as desired. Paradise includes facilities for controlling access to sharable VSMs above and beyond those provided by the host computer systems. Thus results of computations can be made available to "consumers" without compromising the security of the computers carrying out the computation.

Paradise includes a sophisticated fault-tolerance facility that allows programs to guarantee data integrity despite hardware failures, abnormal process termination, lost connections between computers, and other failures. It is structured around a begin-commit transaction strategy similar to that used by state-of-art database systems.

Thus, the Paradise tool and conceptual model can serve both to enrich the developer's coordination tool set for parallelism AND to enable integration into a distributed systems ensemble.