

Parallel Acceleration System (PAS™)

Accelerates the Development of Parallel Sensor-Processing Applications

- Scalable to 1000+ processors
- Maximizes resource utilization to achieve ultimate performance
- Improves developer productivity for image and signal processing
- Complete application host and device coverage
- Reduces code size and complexity



The Parallel Acceleration System (PAS™) from Mercury Computer Systems is a standards-based multiprocessor communication library that accelerates the development of parallel sensor processing applications, while maximizing the performance of inter-processor data communications and facilitating software portability. Using the Data Reorganization Interface (DRI) standard, PAS software provides high-performance data reorganization concurrent with data communication, enabling applications to maximize resource utilization and achieve nearly maximum machine speeds.

The PAS library encapsulates years of experience with multicomputer systems into a multiprocessor communication library that is fast, productive, and easy to use. For developers of parallel processing applications ranging from medical imaging to large-scale defense signal processing, the PAS library reduces development time, lines of code, and ongoing maintenance costs. The PAS application programming interface (API) accelerates the development of parallel sensor-processing applications while maximizing the performance of inter-processor data communications.

Typical Multicomputer Processing

Multicomputer processing can be defined as a collection of cooperating processes executing on multiple processors. A typical multicomputer application divides data and tasks among available processors and processes. The application must then manage flow control, event control, latency, communication, and computation. When building a system solution, it is necessary to integrate host and compute element (CE) processors along with I/O and memory devices.

These tasks are tedious and complex to build, and they have a profound impact on the final application. PAS is designed to help simplify all these tasks.

Productivity and Ease of Use

The PAS library was developed by a team of industry experts to reduce the complexity and effort of implementing parallel multiprocessor communications and data reorganization. Developers can learn to use PAS quickly, because it handles data in a way that supports the user's view of the data in terms of the shape of the data to be moved and the element type: vector data as bytes, matrix data as integers, data cube data as float, and N-dimensional data as either interleaved or split complex. This unique feature increases the conceptual understanding of the completed application by putting the data into terminology the user understands. With the inherent simplicity and ease of use of the PAS library, the amount of communication and data reorganization code can often be reduced by as much as 80%.

PAS Network

A PAS network is not a network of threads, but rather a tightly coupled network of processes and memory buffers running on a number of compute elements. Each PAS memory buffer is accessible by every process in the PAS network. A set of barriers and a set of semaphores reside within each memory buffer for managing data and events. A PAS heap also resides within each memory buffer for making distributed allocations, which can be accessed by all local and remote processes.

PAS Process Sets

The PAS API supports the concept of process sets, which are subsets of the processes in the PAS network. The processes within a PAS process set typically perform the same function on separate portions of the distributed data. Each PAS process usually runs on its own CE for performance considerations, but PAS allows multiple processes to run on the same CE and multiple threads within each process.

PAS process sets provide scalable targets for data distribution and multi-point data transfers, synchronization, and signal processing algorithms. A process can belong to more than one process set; for example, a process could participate in the processing of multiple data-parallel stages by belonging to all the corresponding process sets. Changes to processor distribution and the number of processors used are made parametrically at run time without the need for re-coding, recompiling, or even restarting the application.

Distributed Memory Buffers

In addition to allocating a block of memory (a buffer) associated with a single process, the PAS library provides a means of allocating distributed buffers that span all the processes in a process set. This makes working with distributed buffers as easy as working with single-process buffers.

Figure 1 shows the memory buffer allocations for a three-dimensional cube distributed over four processes. The allocation functions return a PAS partitioned buffer object that acts like a single reference to the entire distributed buffer. Because data allocation is usually based on process sets and process sets are defined dynamically, the application can scale dynamically with respect to available processors and problem size.

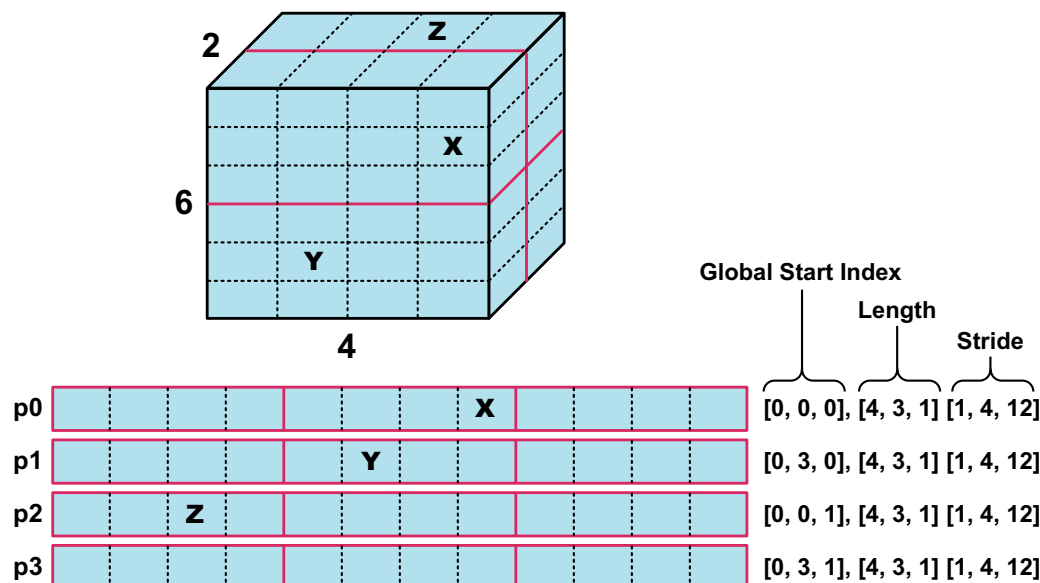


Figure 1. Distributed memory buffers

High-Performance Data Movement

The PAS library has the ability to move and reorganize data as a group while meeting or exceeding the most stringent latency requirements. It optimizes the performance of interprocessor data transfers in large systems and provides a means of synchronizing execution of distributed processes. With support for early binding of interprocessor data communications, PAS minimizes data communication overhead during tight inner-loop operations.

The PAS library makes the best use of underlying connection fabric hardware and DMA engines to achieve optimum performance. The PAS library can move data at nearly machine speeds with a dramatic reduction in the code and complexity and with numerous additional features.

Data Reorganization

The PAS API contains rich data reorganization capabilities based on those defined in the Data Reorganization Interface (DRI) standard. DRI is a software interface for performing data-parallel distribution and reorganization operations such as transpose and reshape, which are frequently required in scalable, high-performance, embedded computing applications. DRI provides increased ease of use compared to point-to-point middleware by providing abstractions for multi-dimensional datasets, partitioning and distribution methods such as block, block-cyclic, overlapped elements, and a high-level interface that frees applications from having to orchestrate the multitude of individual transfers required in a single data reorganization.

The multi-buffering semantics of DRI enable the application to overlap communication and computation. With DRI implemented in PAS, users gain the benefits of both high performance and a standard API. The PAS multi-point data reorganization and multi-buffering functions are tuned specifically for multiprocessors. PAS automatically handles large data reorganization operations such as multiprocessor corner turns, relieving the user from having to handle the complexity of these operations.

The PAS library provides sophisticated functions for N-dimensional data reorganizations. These functions can carry out a diverse set of partitioned and non-partitioned data transfer operations, as shown in Figure 2.

PAS Extensions to DRI

Some high-performance emerging applications require features not currently defined by the DRI standard. Most notable of these is dynamic management of data sizes, process sets, and memory. The PAS library supports the following extensions to the DRI standard:

- Support for the split-complex data type
- Dynamic buffer utilization for efficient memory utilization
- Workflow management
- Integration of I/O and memory-only devices through a generic interface
- Dynamically changing transfer attributes
- Interrupt-based synchronization and multi-threaded support
- Transferring only a region of interest (ROI) of distributed data

These PAS extensions have been implemented so that users can use the DRI standard API and then pass parameters to the PAS extended functions. This design allows users to maximize the amount of standard code they use and still meet operational requirements.

PAS Channels

A PAS channel, also known as buffered data movement, is an automated pathway between two process sets that frees the user from the complexities of interprocessor communication. The PAS library

supports multi-point, multi-buffered data movement through PAS channels and partitioned buffer sets. Channels save users a great deal of work, while also providing dynamic buffer utilization, leaving more memory for application code and data.

Tight integration of I/O and memory-only devices is provided directly for Mercury devices or through a generic interface for third-party devices. The generic interface essentially allows the user to write a PAS channel driver. Channels support all these devices as endpoints, allowing efficient communications to and from attached devices.

When data and processing methodologies are relatively consistent, static allocation of resources works well. When data is not consistent and/or processing methods change, then users need dynamic control over the application and data. PAS channels support dynamically changing transfer attributes, such as data size, shape, multi-processor distribution, and user-supplied meta-data for applications with multiple modes.

Efficient use of vector instructions such as the AltiVec™ unit in the PowerPC® G4 processor requires processing complex data in a split form rather than an interleaved form. The split form separates the real and imaginary data into separate arrays. Data transfers involving different data formats can be difficult, but PAS channels simplify these operations. By defining the data on either side or both sides of the channel as complex and split, the user can automatically transfer data from interleaved complex to split complex, split complex to split complex, and split complex to interleaved.

Workflow Management

Typically in signal processing applications, more than one process or set of processes ends up doing the same work on different datasets. Workflow management can help in this environment, through round-robin and next-available multi-buffered, multi-point data transfer policies. Round robin is suitable when the amount of processing performed by each destination process set is similar. When the amount of processing performed by the source or destination is unpredictable, depending on the data, then next-available is the appropriate policy. Aggregate channels provide a framework to support workflow management.

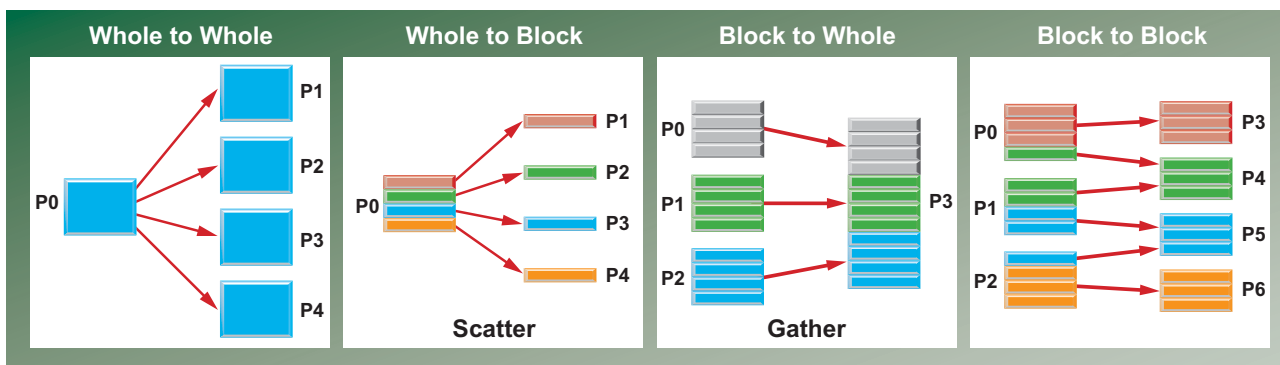


Figure 2. Data reorganization operations

Synchronization

PAS synchronization is accomplished using semaphores; barrier synchronization; and channels, which are defined over process sets. Both non-blocking and blocking synchronization are supported in semaphores and channels, and are under user control. With PAS channels, synchronization is completely managed by the channel, freeing users from the complexity of managing their own data transfers.

Scalability

Multicomputing applications often need to scale the number of processing elements from one to hundreds of processors, as users need to reduce their software development time and expense. The PAS library offers a software API specifically designed for scalable, distributed-memory multiprocessor applications. Applications developed with PAS can be made to scale over a different number of processors simply by changing a run-time variable, representing the number of processors. Definition of process sets and assignments to specific processors is also done at run time without the need for recompilation. PAS has been proven to scale to 1,000 processors.

PAS and Challenges Drive Innovation are trademarks of Mercury Computer Systems, Inc. Other products mentioned may be trademarks or registered trademarks of their respective holders. Mercury Computer Systems, Inc. believes this information is accurate as of its publication date and is not responsible for any inadvertent errors. The information contained herein is subject to change without notice.

Copyright © 2006 Mercury Computer Systems, Inc.

510.00E-0206-DS-PAS



Corporate Headquarters

199 Riverneck Road
Chelmsford, MA 01824-2820 USA
+1 (978) 967-1401 • +1 (866) 627-6951
Fax +1 (978) 256-3599
www.mc.com

Worldwide Locations

Mercury Computer Systems has R&D, support and sales locations in France, Germany, Japan, the United Kingdom and the United States.

For office locations and contact information, please call the corporate headquarters or visit our Web site at www.mc.com.