SCCS 710

The High Performance Switch and Programming Interfaces on IBM SP2 (*draft*)

Gang Cheng, Marek Podgorny
(*gcheng,marek@npac.syr.edu*)

Northeast Parallel Architectures Center
Syracuse University

Abstract

This is a review report about the high performance switch and the programming interfaces on IBM SP2 . Based on a collection of materials from many other sources and our experience using SP2, we summarize internals of HPS configuration and its supporting software, transport protocols, message passing libraries, programming interfaces and software layers. Performance data of the HPS are also collected and given in this report to help us develop strategies of benchmarking the SP2 communication subsystems.

1. Introduction

High performance switch (HPS) is the central component of IBM 9076 SP2 system, which enables the high-bandwidth and low-latency communication subsystem on SP2. SP2 system is a relatively new product from IBM and descriptions about its HPS are currently not available in the usual manuals and are scattered in different places in the forms of technical reports, research papers, and WWW web pages. It is our intention in this paper to summarize the HPS and the programming interfaces built on it, based on our understanding and experience of using SP2, and collections of materials from other references. We describe internals of HPS hardware configuration and its supporting software, transport protocols, message passing libraries, programming interfaces and software layers. This is a review report whose major purpose is to provide as informative as possible materials for software developers to have a general yet detailed picture about HPS and parallel programming environments available on SP2. This report will also serve as a base understanding of SP2 for new SP2 application programmers and software developers who want to know more about HPS internals.

Because one of our major motivatons for this review is to understand the performance issues of HPS and SP2's commuinication subsystem, we also include collections of performance data from many other sources to provide with a performance picture of the switch. This document will also be used for NPAC to design technical strategies of benchmarking the SP2 communication subsystems.

2. Background information about HPS on SP1 and SP2

IBM SPx has experienced several significant upgrades during the past two years since its introduction to the market in early 1993. You may notice different terminology which are sometimes confusing as they may only be valid to one of the SPx platforms.

SP1: EUI(TB0),EUIH(also referred as MPL/p),HPS Adapter-1
SP2: MPL,HPS Adapter-2 (TB2),User Space(uCSSCI)

(1) EUI on SP1
EUI is the reference name of the transport protocol on HPS of SP1. It was also the mnemonic for the first release of MPL that was available on the SP1. EUI-H is an experimental upgrade of EUI that later underwent a name change to become MPL in the SP2 environment. EUI-H was also called the light-weight or lightspeed EUI and is a research tool which was later discontinued by IBM on SP2. HPS Adapter-1 was shipped with SP1.

(1) MPL on SP2
MPL (Message Passing Library) is the component of the IBM AIX Parallel Environment that enables the user to write parallel applications. As its name indicates, the MPL routines enable message passing between individual tasks of a parallel application for communication of data and synchronization of the tasks operations. The enhanced HPS Adapter-2 is new to SP2 system. The two communications modes supported on SP2 and in MPL are AIX sockets with TCP/IP, and a new high performance user space library.

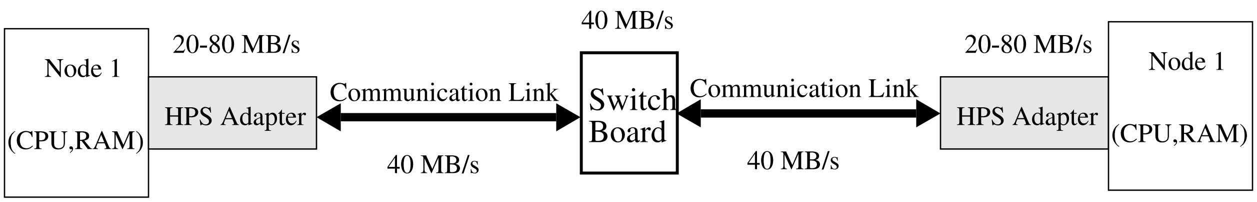
We will describe the major differences between HPS on SP2 and SP1, and the transport protocols supported in later sections.

3. Hardware Configuration Features of HPS on SP2

HPS on SP2 provides the internal(hardwire) message passing fabric that connects all of the SP processors together to allow any-to-any internode connection and all processors to send messages simultaneously. It is a packet-switched network (vs. circuit-switched) and a multistage network which can add switches as system is scaled upward. The peak bandwidth of overall HPS is 40 MB/s in bi-direction and about 1 μs latency for point-to-point data transfer. It supports for multi-user environment such that multiple jobs may run simultaneously over the switch (one user does not monopolize switch). With built-in error detection, it supports path redundancy that permits routine to be generated even when there are faulty components (eg. nodes) in the system.

When HPS on SP2 is referred in general, we are actually talking about three major components of the HPS: Switch Adapters, Communication Links, and a Switch Board, as illustrated in Figure 1 for the configuration of two nodes. They perform different functions in node-network and network-network(switching) communications. They have different peak bandwidths and constraints due to electric or physical configuration limitations. The overall 40 MB/s HPS peak bandwidth is the minimum one(s) of the three, namely, the communication links and the Switch board.

Figure 1. Diagram of HPS components connecting two nodes



Communication Link:

All communication in SP2 uses point-to-point bidirectional communication links that consist of two channels, each carrying data in opposite directions to their respective output port and input port pairs. For each channel, the actual transmission is via a set of 10 signal lines, 8 for data, one for tag and one for the token. A major enhancement to the SP2 Switch is the incorporation of clock distribution into the switch communication links. In the SP1 Switch, clock distribution is accomplished via a separate clock redrive network. In SP2, each link is additionally capable of carrying redriven clock signals in both directions.

Switch Adapters:

The SP1/2 switch adapter card connects a processor's Micro Channel to (both) output and input ports of the switch network via an ASIC chip called the MSMU (Memory & Switch Management Unit). The MSMU provides communication services commonly found in the physical, datalink and network layers of most communication adapters. The MSMU has one output and one input port implementing the link level protocol of the switch.

There have been two adapter versions: HPS Adapter-1 (20 MB/s), HPS Adapter-2 (80MB/s). One adapter per SP node.

(1) HPS Adapter-1

HPS Adapter-1 was developed for SP1 with a simple configuration for quick time to market. In the SP1 adapter implementation, the MSMU appears as a memory mapped device to the processor. Switch packets are moved in and out of the processor node via the memory mapped MSMU FIFO’s. The SP1 adapter has no DMA capability. It is the node processor’s responsibility to move switch packets in and out of the MSMU FIFO’s using PIO instructions.

The HPS Adapter-1 did not use all of the bandwidth of the comunication links, due to the poor performance built in the node-to-network interface of the adapter. Lacking local intelligence, it relied on the host processor to transfer data to and from memory and perform all communications processing in a bus-slave fasion [7] which limited the bandwidth to a peak Micro Channel transfer rate of 20 MB/s. The reported 8.3 MB/s bandwidth and 28 usec latency using MPL/p(EUI-H) is largely due to the HPS Adapter-1 limitation.

(2) HPS Adapter-2

As shown in Figure 2, HPS Adapter-2 has enhanced configuration over the previous implementaion on SP1 to improve bandwidth and offload communication tasks from the node processor. It uses the Micro Channel's bus master and streaming capabilities for a 80 MB/s peak rate. In addition, it incorporates an Intel XR 64-bit microprocessor, with 8 MB of four-way interleaved DRAM, for communication coprocessing. Finally, the SP2 implementatioon moves data checking (in particular, CRC generation) into hardware.

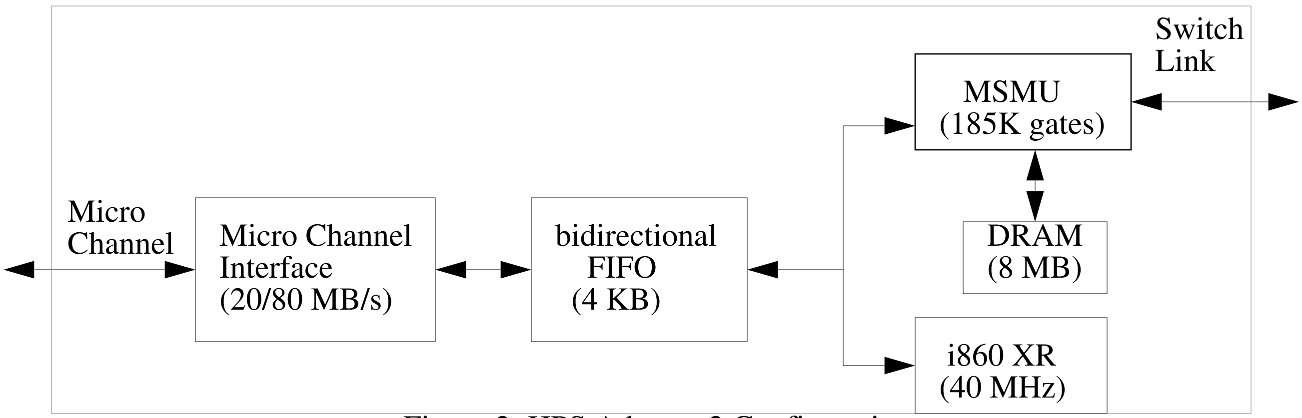


Figure 2: HPS Adapter-2 Configuration

On the SP2 adapter, the interface to the Switch network and the adapter memory is the MSMU, and the interface to the MSMU is through the i860 bus. Thus data transfers involve two buses -- the Micro Channel and the i860 bus. The Adapter joins these buses with a 64-bit-wide, 4KB bidirectional FIFO. Each interbus data transfer then has two componets: Micro Channel to/from FIFO, and i860 bus to/from FIFO. This configuration feature maximize flexibility and performance. The former bus supports all Micro Channel transfer rate up to 80 MB/s, while the latter makes use of i860 bus pipelining to achieve 160 MB/s. The adapter also provides a bidirectional FIFO bypass that allows the node processor to address the i860 bus directly using PIO instructions. This permits the MSMU to be addressed directly, as in the HPS Adapter-1; it also gives the RS/6000 access to adapter memory.

These new features gave SP2 approximately five times the long message bandwidth of SP1 with no modification of the switching network. unidirectional bandwidth is about 35 MB/s for Power2 nodes. This bandwidth is equivalent to the communication link capacity, considering packet overheads such as error checking, sequencing, and route flits. The bidirectional bandwidth ranges up to 48 MB/s, constrained mostly by Micro Channel degradation for short transfers. In addition to increasing bandwidth, offloading data transfer frees the RS/6000 to perform computation. On the "wide" SP2 nodes, about 25% of the processor reminds available during bidirectional communication, and about 40% during sends, though not all applications can expoit this.

Switch board:

A switch board contains 8 logical switch chips with 16 physical chips for reliability reasons. One switch board per SP frame. The switch operates at 40 MHz, providing peak bandwidth of 40 MB/s over both byte-wide channels of each communication link. It has 500 ns hardware latency.

SP2 networks are bidirectional multistage interconnection networks, in which each communication link comprises two channels which carry data inopposite direction. For all SP configurations, there are at least FOUR usable paths between each pair of nodes, except for node pairs that are directly attached to the same switch element.

The switching element of the board is the Vulcan switch chip [7]. As shown in Figure 3, each switch chip contains 8 receiver modules and 8 transmitter modules, an unbuffered crossbar, and the central queue.

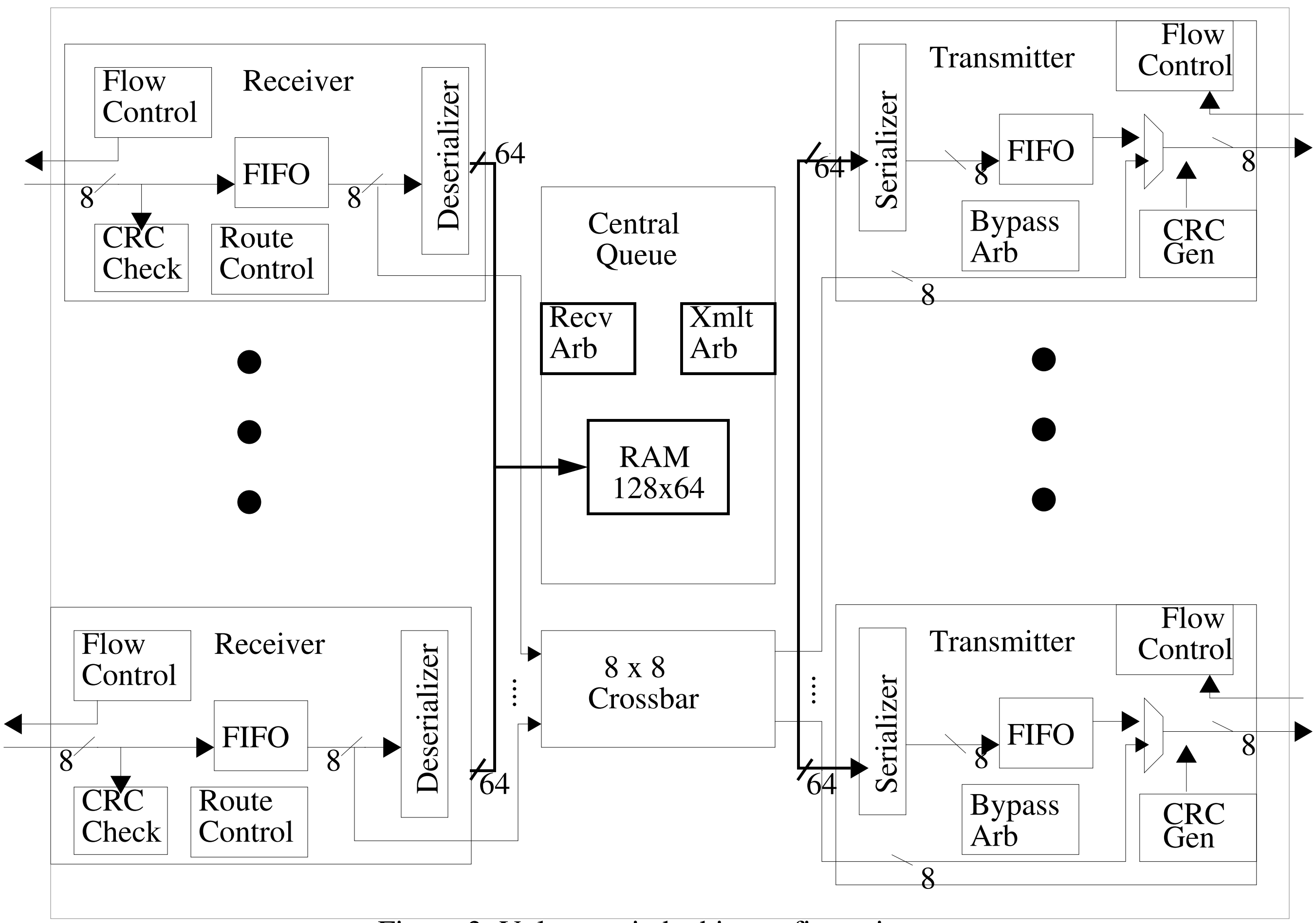


Figure 3: Vulcan switch chip configuration

Transmitters/Receivers perform five major functions:
(1) administering the link flow control protocol
(2) computing/checking the link CRC codes
(3) buffering outgoing/incoming data in FIFO
(4) decoding (Receiver only) packets routine information
(5) serializing outgoing /deserializing incoming packets when the packet is blocked

Crossbar is an unbuffered flit-serial logical crossbar that allows packets to pass directly from the receivers to the transmitters, whenever there is no contention for the transmitter output port, otherwise, the packets recieved from the reciever input ports are buffered in the central queue.

The central queue stores packets until they can be transmitted. The storage, a 128 by 64-bit dual-port RAM, holds up to 128 eight-flit packet chunks. The queuedoes not reserve a fixed amount of space for each output port; storage is allocated dynamically based on demand. Stored packets are queued in FIFO order on eight linked lists, one list corresponding to each of the eight switch output ports.

4. Communication Protocols on HPS

There has been three different communication protocols over HPS:

(1) IP (Internet Protocol)

IP communications is the default mode because it allows shared usage of an adapter by multiple processes on a node. Any application may communicate over HPS by opening a standard AIX socket and specifying appropriate IP addresses. UDP/IP calls are made at the kernel level supporting IP communications through HPS adapters. Additional system facilities based on IP, such as NFS,AFS, may also be configured to use the HPS. With the possible exception of application that depend on network-specific functions (LAN broadcasts, for example), most applications work over the HPS without modification.

In the case of IP communication on HPS Adapter-2, the i860 is used to only initiate DMA between the network and RS/6000 storage. Usual system buffer and device driver in UNIX kernel, several layers of IP protocol are gone through to faciliate the IP-based comunication, which results in higher transport latency.

(2) uCSSCI (or User Space)

The user space communication mode, uCSSCI, is intended for parallel applications that require maximum communications performance, lowest latency and highest bandwidth. To achieve higher performance, however, a node must be dedicated to one parallel application at a time. Only a single AIX process on a node may use the user space communication library, and nodes must be allocated to the parallel application through the Resource Manager by the Parallel Environment (including MPL,POE,pdbx) or PVMe. The Resource Manager should group nodes into named resource pools to separated user space nodes from shared usage nodes (this is configurated by your SP2 system administrator).

High performance in terms of both latency and bandwidth can only be achieved if the UNIX operating system can be bypassed such that the entire communication stack is executed in user space. This is major motivation of the "User Space". However, with this implementation, the HPS adapter has to be dedicated to one process per node.

This transport protocol is structured in three layers, as shown in Figure 4:

Message Layer (eg. MPL,MPI,PVMe)
Pipe Layer
Packet Layer

Figure 4: communication stack of the "user space" protocol

The Packet Layer provides point-to-point packet transport facility among processors by direct interaction with the communication network via the adapters. This layer creates packets, and inserts appropriate routing information into each of the generated packets.Analogous to IP-UDP, this layer is not assumed to provide reliable transport or packet ordering.

The Pipe Layer provides a reliable, flow controlled, byte-stream oriented communication layer. Each processor maintains a send and a receive buffer (called pipe) to every other processor in the parallel job. These buffers perform functions similar to UNIX pipes and sockets, but they are placed in user space to avoid the cost of a UNIX kernel access. This layer uses the typical mechanisms, such as flow control, acknowledgments and retry after time-out etc.

The Message Layer is a set of message passing libraries consists of a number of functions for point-to-point communication, collective communication, and environmental inquiry and management. There are currently three such message passing libraries: MPL, MPI-H and PVMe. We will describe them in later sections.

(3) EUI-H

This is a lightspeed version of the EUI on SP1. It is later discontinued by IBM on the SP2 platforms. It is the predecessor of the protocol in above (2) on SP1 switch adaptor.

The major performance difference between EUIH and EUI is the improved message-passing latency, as described in [4].

5. Communication Subsystem Support (CSS)

The CSS component of AIX PSSP contains HPS adapter diagnostics (both Adapter-1 and Adapter-2), switch initialization and fault handling software, device drive and configuration methods(config/unconfig) and parallel applications programming interfaces. At boot time, CSS configures the HPS adapter including execution of a Power On Self Test (POST Diagnostics). In addition to the POST Diagnostics, a comprehensive set of adapter diagnostics is available on-line.

CSS perform switch initialization as part of system startup and automatic switch reconfiguration following a switch fault. SP2 nodes or frames which are not operable are removed from the configuration and reported through the System Data Repository to the Resource Manager and SP System Monitor. Most faults and detected errors are successfully retried transparently to an application.

Base support for parallel application task communications is provided by CSS and its Communications Interface (CSS_CI) component. Two communication libraries are provided by CSS-CI: network based and user based.

In the network based (or IP) library, UDP/IP calls are made at the kernel level supporting IP communications through Ethernet or HPS adapters.
The user based (or user space) native communication library maximizes performance of parallel applications since direct communications occur between the application and the SP2 HPS. No kernel calls are made. The IBM AIX Parallel Environment (including MPL,POE,VT and pdbx) and PVMe directly support this approach. Use of CSS-CI for both IP and user based communications provides improved application reliability through full message retry and significantly improved performance through low latency task to task communications (less than 40 microsecond) on both switch adapters. Communications bandwidth improvements are also possible through the HPS Adapter-2 (> 30 MB/s).

6. Programming Interfaces on SP2

There are three layers of programming interfaces for message-passing on SP2 using HPS.

(1) the lowest high performance level

MPL is the native message-passing library provided by the vendor. Like similar native libraries on other parallel machines, it is the most flexible and highest performance layer to support explicit message passing on SP2 and HPS. Its sole purpose is to exploit the maximum bandwidth of HPS in a concurrent program, while portability issues to other platforms are not considered. Notice that MPL can still have some kind of portability in a sense that one programming interface can support three different transport protocols (IP on non HPS, IP on HPS, Use Space on HPS) on SP2 with Ethernet,token ring or HPS or a IBM RS/6000 workstation cluster. For applications requiring maximum communication performance, it is recommended to use MPL with User Space option.

(2) the portable intermediate level

Standard message passing interfaces are developed either by the vendor or by third parties with the major purpose of achieving highly portable programs across multiple parallel platforms. Major software of this kind are:

MPI - message-passing interface specified by MPI Forum . There are currently two implementations on SP2. MPI-F from IBM and MPI-CH from ANL/MSU.

PVM - Parallel Virtual Machine. PVMe is the IBM implementation of PVM originally developed at ORNL/UTK. It can run under the three protocols.

P4 - a portable message-passing package from ANL.

Chameleon - another lightweight, portable message-passing system from ANL. It is built on MPL, MPI or P4.

Because the portability and a generic message-passing framework is the first consideration, it has more or less performance penalty when ported to a specific platform.

(3) the highest level

HPF - a high level data parallel programming language. It uses the data-parallel model on the MIMD architecture to achieve concurrent programs with high portable, ease of programming, and implicit message-passing in data parallelism. It often has to sacrifice some degree of performance thus unable to be suitable for general applications and for building software tools.

Fortran-M - a programming framework to support task parallelism and modular programming paradigm in sequential and parallel Fortran programs.

7. MPL (Message Passing Library) on SP2

MPL is the lowest level of parallel message passing library provided by IBM SP2 as a programming interface for converting a serial Fortran77, C or C++ program into a parallel application on SP2 via subroutine calls. The MPL provides a rich and diverse set of subroutines for coding simple operations, such as task-to-task message passing, as well as the more advanced operations required for highly complex communications.

At either compilation or execution, the user specifies whether parallel task communications in MPL can occur via the IP protocol through the Ethernet or HPS adapters (if configured for IP) or via the User Space access to the HPS for maximum performance. Based on user's selection, the POE will link (statistically at compile time or dynamically at runtime) different communication libraries for the MPL program.

Like most commonly used message-passing libraries such as MPI,PVM,P4,CMMD, MPL contains the following major communication primitives:

. point to point message passing library
. collective communications library (CCL)
. synchronous and asynchronous communications
. configuration, control and management
. exception handling

MPL (formerly EUI) is IBM's message-passing interface to the HPS. POE is required as the parallel operating environment to control running of MPL programs. Any high-level programming interface that make use of Switch/Us uses POE.

MPL tasks can communicate with each other using either the Internet Protocol (IP) library or with the User Space (US) library. Some differences between these two libraries include: IP communications can occur over ethernet or the high performance switch. US communications can only be conducted over the high performance switch. Only one US task can use the switch adapter on a node. When the task starts, it reserves (dedicates) the adapter to that task only. No other US tasks can use the adapter for the duration of the task which dedicated the adapter. IP permits multiple tasks using IP communications to share the use of the switch adapter on a node simultaneously. Additionally, IP will share the adapter with a US task. IP bandwidth is approximately 7-8 MB/s. US bandwidth is approximately 31-35 MB/s. IP is recommended for the interactive nodes where multiple users are using a node at the same time. US is recommended when nodes are allocated by the batch system or when nodes have been reserved for users performing benchmarking.

IP communications over the high performance switch have a bandwidth of approximately 7-8 MB/s. US communications over the high performance switch have a bandwidth of approximately 31-35 MB/s. Bandwidth may be affected by competition from other users.

8. PVMe

PVMe is a product from IBM which enables PVM users to exploit the User Space communications library for the SP2 high performance switch. This means that PVM codes can approach the performance of MPL codes which run US over the switch. PVM codes do not need to be rewritten, or even recompiled, simply linked to the PVMe libraries instead of the usual PVM libraries.

Although public domain (ORNL) PVM can communicate over the SP2 high performance switch, it can not take advantage of the SP2 User Space communication library. PVM must perform communication by IP protocol. There is a significant difference in bandwidth between these two methods. MPL jobs running with US over the high performance switch can achieve throughput in the 31-35 MB/s range. PVM running IP over the switch will achieve between 1-2 MB/s. PVM running IP over ethernet is usually half (or less) of this.

9. MPI

MPI is a message-passing library interface specification (not a specific implementation), and was designed by a broad-based group of parallel computer vendors, and contains features of other systems. MPI will eventually be an IBM product (also from other vendors). There are currently two efficient (use MPL) implementations of MPI available on SPx: IBM Research (MPI-F) and ANL/MSU (MPICH).

MPI-F is an experimental IBM implementation of MPI which works with POE/MPL system. It supports transport layers of Switch/IP and switch/US. (Ethernet/IP is not supported).

MPI-F has much richer point-to-point message passing layer (total 53 functions) than the native MPL (32 functions). While the point-to-point functions for user space are directly implemented on the pipe and packet layers described in Section 4, higher level of MPI functionality (i.e., collective communication, topology, etc.) are implemented on top of MPI point-to-point message passing.

The major functional difference between MPI and MPL is the introduction of communicators and buffering mechanism in MPI.

10. Performance Results and Existing Benchmarks for SP2

Table 1 below is a collection of the performance data of HPS reported by IBM and others.

Table 1: Performance Data of the HPS on SPx

Source	Communication Protocol	Message Passing Interface	SPx Platform	**Minimum Latency (usec)**	**Peak Point-to-Point Bandwidth (MBytes/s)**
IBM [3]	User Space	unknown	SP2	< 40	> 30
IBM [4]	User Space	unknown	SP2	-	31-35
IBM [13]	User Space	unknown	SP2	39	35.54
ANL [1]	User Space	unknown	SP2	63	35
NAS [5]	User Space	MPL	SP2	46	34
NAS [5]	User Space	MPI-F	SP2	46	34
NAS [5]	User Space	MPICH	SP2	62	34
NAS [5]	User Space	PVMe	SP2	83	31
USC [13]	User Space	MPL	SP2	46	35.54
USC [13]	User Space	MPI	SP2	67	35.54
USC [13]	IP	unknown	SP2	600	-
ANL [1]	IP	unknown	SP2	-	7 - 8
IBM [4]	IP	unknown	SP2	-	7 - 8
IBM [5]	EUI-H	MPL/p	SP1	30	8.5

The closed formulae given by IBM and observed by Z. Xu at USC [13] for point-to-point communication estimation are (in usec.): *46+0.035M* (USC) and *39+0.028M* (IBM), where M is the message length in bytes.

[13] has done a complete and thorough benchmarking for collective communication operations using MPL and MPI primitives, as listed in Table 2-4 below, where M is the message length in bytes, and N is the number of SP2 processor nodes involved in the communication.

Table 2: Collective Communication Performance in MPICH and MPL

Operation	MPL Timing Formula (*u*sec)	MPICH Timing Formula (υsec)
Broadcast	$(16 \log N + 10) + (0.025 \log N)M$	$(40 \log N + 20) + (0.037 \log N)M$
Gather	$(17 \log N + 15) + (0.025N - 0.02)M$	$(24N + 84) + (0.045N)M$
Scatter	$(17 \log N + 15) + (0.025N - 0.02)M$	$(24N + 105) + (0.026N + 0.03)M$
Shift	$(6\log N + 60) + (0.003\log N + 0.04)M$	-
Total Exchange (index)	$80\log N + (0.03N^{1.29})M$	$(125N - 22) + (0.06N^{1.29})M$
Barrier	$94\log N + 10$	-
Reduce	$20\log N + 23$	-
Prefix	$60\log N - 25$	-

Table 3: Performance Estimation for MPL Collective Communication

Operation	Latency (usec)	Asymptotic Bandwidth (MBytes/s)	Aggregated Asymptotic Bandwidth (MBytes/s)
Broadcast	$16 \log N + 10$	$40/\log N$	$40N/\log N$
Gather/Scatter	$17 \log N + 15$	$1/(0.025N + 0.03)$	$N/(0.025N + 0.03)$
Shift	$6\log N + 60$	$1/(0.003N + 0.04)$	$N/(0.003N + 0.04)$
Total Exchange	$80\log N$	$33.3N^{-1.29}$	$33.3N^{0.71}$

Table 4: Performance Estimation for MPI Collective Communication

Operation	Latency (usec)	Asymptotic Bandwidth (MBytes/s)	Aggregated Asymptotic Bandwidth (MBytes/s)
Broadcast	$40 \log N + 20$	$27/\log N$	$40N/\log N$
Gather/Scatter	$24N + 84$	$22/N$	22
Shift	$24N + 105$	$1/(0.026N + 0.03)$	$N/(0.026N + 0.03)$
Total Exchange	$125N$-22	$16.7N^{-1.29}$	$16.7N^{0.71}$

In additon, NAS benchmarks for SP2 are available [5], which includes:

. Kernel benchmarks
  . Embarrassingly parallel
  . 3D FFT PDE
  . Integer Sort
  . Conjugate Gradient
  . Multigrid

. Simulated Application Benchmarks
  . LU decomposition
  . Scalar pentadiagonal
  . Block tridiagonal

11. Methodologies for Benchmarking HPS and SP2 Communication Subsystem

Based on the above introduction and the message-passing programming interfaces available on SP2, there are four major benchmarking classifications:

(1) benchmarking at different programming levels

Different levels of programming interfaces present to the end-user or software developer with different programming capabilities in ease of programming, problem expressiveness and declaration, flexibility of control. There are always some trade-offs among implementation performance, programming ease and portability. For a system software developer in areas such as data-mining, choose of message passing libraries on SP2 should weight all the factors in the specific development of the system software.

From performance benchmarking point of view, there are two levels of message-passing libraries on SP2 must be considered:

. the native message-passing library (MPL)

This is the lowest level of abstraction of a programmable SP2 provided by the vendor. It should exploit the maximum performance of the underlying hardware and give the highest flexible control over the SP2 system, though less expressive.

. the portable explicit message passing interfaces

Besides the native one, more and more vendors start to support one or more portable message passing interfaces on their machines, such as MPI,PVM,Linda,Express. In additional, there are active implementations from third parties of those portable libraries on different platforms. It is preferable to choose the implementation of the vendor as it should have higher performance value, while the programming interface should be identical. On SP2, MPI-F and PVMe are the two choices.

(2) benchmarking different communication primitives

Basically, there are two types of communication services in a message-passing library: point-to-point and collective.

Point-to-point communications:

It involves two communicating processes in the form of various modes of SEND and RECEIVE operations, which may fall into the following categories:

. synchronous and asynchronous send/receive (some called blocking/non-blocking)
. synchronous and asynchronous data exchange (e.g., swap,shift,scan,etc)
. contiguous and non-contiguous vector data

Collective communications:

It involves a group of processors to collective perform an atomic (internal synchronized) group operations, such as broadcast, gather, scatter, reduction, multicast, and all-to-all broadcast). Collective communication consists of three subclass: one-to-all, all-to-one, and all-to-all

(3) benchmarking with different performance-related parameters

. message size,buffer size,number of buffers(or messages)
. machine size
. latency (message size=0) v.s. bandwidth
. number of primitive calls
. number of repeated iterations, maximum/minimum and average measurements
. protocols (IP/Ethernet,IP/Switch,US/Switch,Wide/Thin nodes,etc)

(4) benchmarking application-specific communication patterns

In most situations, a specific application can find some dominant and typical communication patterns in their message-passing operations. For example, a sort-oriented application using merge-sort operators may require frequent run-time data redistribution on partially sorted data in arbitrary number of dimensions and a given data decomposition scheme. This may need to benchmark the algorithms and techniques developed to minimize the amount of data exchange.

With the specific data-mining applications, certain operations can be benchmarked before the actual development effort to provide performance input for the system architects to compare with and make decisions to different algorithmic approaches, while they can be implemented using the performance results from (1) - (3) above.

References

[1] W. Gropp and E. Lusk, "Users Guide for the ANL IBM SPx", ANL/MCS-TM-199, Dec., 1994.

[2] Web pages from Cornell Theory Center (CTC) "http://www.tc.cornell.edu", May, 1995

[3] Web pages from IBM POWERparallel Systems Server "http://lscftp.kgn.ibm.com:80/pps/", May, 1995

[3] Web pages from Argonne National Laboratory (ANL) "http://www.mcs.anl.gov/", May, 1995

[4] Web pages from Maui High Performance Computing Center "http://www.mhpcc.edu mhpcc.html",May, 1995

[5] Web pages from Numerical Aerodynamic Simulation Facility (NAS) "http://lovelace.nas.nasa.gov/Parallel/SP2/nas_sp2_home.html", May, 1995.

[6] P. Hochschild, "EUIH: An Experimental EUI Implementation (preliminary) Version 1.06.3), Internal Implementation Notes, IBM T. J. Watson Research Center, Sept., 1993.

[7] C. B. Stunkel, D. G. Shea, et al., "The SP2 Communication Subsystem", Technical Report, IBM T.J. Watson Research Center, August, 1994.

[8]. C. B. Stunkel, D. G. Shea, et al., "The SP1 High Performance Switch," in Proc. 1994 Scalable High-Performance Computing Conference, pp. 150-157, May 1994

[9] IBM 9076 Scalable POWERparallel Systems, SP2 Administration Guide SH26-2486-01, 1995.

[10] V. Bala, J. Bruck et al., "CCL: A Portable and Tunable Collective Comunication Library for Scalable Parallel Computers," IBM Research Report. RJ-9284, April 1993.

[11] V. Bala et al., "The IBM External User Interface For Scalable Parallel Systems," Parallel Computing 20 (1994), pp. 445-462.

[12] H. Franke, P. Hochschild et al., "An Efficient Implementation of MPI," in Proc. of Programming Environment for Massively Parallel Distributed Systems, Monte Verita, Switzerland, 1994.

[13] Z. Xu and K. Hwang, "Modeling Communication Overhead: MPI and MPL Performance on the IBM SP2 Multicomputer," (draft) Technical Report, Dept. of EE-Systems, University of Southern California, January 10, 1995.