Appendix A Summary of Communication Operations

This appendix consolidates information about the collective communication operations available in MPI and their time complexities. The times required to complete the operation given in the third column of Table A.1 assume two things:

1. The underlying hardware network has the capability of concurrently transmitting the number of messages given in the fourth column.
2. The underlying network software uses an algorithm that considers the message size and determines the most efficient routing algorithm based on that size.

Operation	**MPI Name**	**Time Requirement**	**Bandwidth Requirement**
one-to-all broadcast	`MPI_Bcast`	$(\lambda + m/\beta)\log p$	$\Theta(1)$
all-to-one reduction	`MPI_Reduce`		
all-to-all broadcast	`MPI_Allgather`		
all-to-all reduction	`MPI_Reduce_scatter`		
all-reduce	`MPI_Allreduce`		
gather	`MPI_Gather`		
scatter	`MPI_Scatter`		
all-to-all personalized	`MPI_Alltoall`		

Table A.1: Collective communication operations in MPI and their complexities, assuming a hypercube-based transmission structure. The time requirement assumes that the physical network supports concurrent transmission of the number of messages given in the last column, which is stated in asymptotic notation. When there is a minimum, listed, it is based on the assumption that the network software will choose the fastest algorithm depending on the message length, m. The parameter λ is the message latency, and the parameter β is the link bandwidth. All links are assumed to have equal bandwidth.

Appendix B MPI Error Values

Symbolic Name	Value	Meaning
`MPI_SUCCESS`	0	Successful return code.
`MPI_ERR_BUFFER`	1	Invalid buffer pointer.
`MPI_ERR_COUNT`	2	Invalid count argument.
`MPI_ERR_TYPE`	3	Invalid datatype argument.
`MPI_ERR_TAG`	4	Invalid tag argument.
`MPI_ERR_COMM`	5	Invalid communicator.
`MPI_ERR_RANK`	6	Invalid rank.
`MPI_ERR_REQUEST`	7	Invalid `MPI_Request` handle.
`MPI_ERR_ROOT`	7	Invalid root.
`MPI_ERR_GROUP`	8	Null group passed to function.
`MPI_ERR_OP`	9	Invalid operation.
`MPI_ERR_TOPOLOGY`	10	Invalid topology.
`MPI_ERR_DIMS`	11	Illegal dimension argument.
`MPI_ERR_ARG`	12	Invalid argument.
`MPI_ERR_UNKNOWN`	13	Unknown error.
`MPI_ERR_TRUNCATE`	14	Message truncated on receive.
`MPI_ERR_OTHER`	15	Other error; use Error_string.
`MPI_ERR_INTERN`	16	Internal error code.
`MPI_ERR_IN_STATUS`	17	Look in status for error value.
`MPI_ERR_PENDING`	18	Pending request.
`MPI_ERR_ACCESS`	19	Permission denied.
`MPI_ERR_AMODE`	20	Unsupported amode passed to open.
`MPI_ERR_ASSERT`	21	Invalid assert.
`MPI_ERR_BAD_FILE`	22	Invalid file name (for example, path name too long).
`MPI_ERR_BASE`	23	Invalid base.
`MPI_ERR_CONVERSION`	24	An error occurred in a user-supplied data-conversion function.
`MPI_ERR_DISP`	25	Invalid displacement.
`MPI_ERR_DUP_DATAREP`	26	Conversion functions could not be registered because a data representation identifier that was already defined was passed to `MPI_REGISTER_DATAREP`.
`MPI_ERR_FILE_EXISTS`	27	File exists.
`MPI_ERR_FILE_IN_USE`	28	File operation could not be completed, as the file is currently open by some process.
`MPI_ERR_FILE`	29	
`MPI_ERR_INFO_KEY`	30	Illegal info key.
`MPI_ERR_INFO_NOKEY`	31	No such key.
`MPI_ERR_INFO_VALUE`	32	Illegal info value.
`MPI_ERR_INFO`	33	Invalid info object.
`MPI_ERR_IO`	34	I/O error.
`MPI_ERR_KEYVAL`	35	Illegal key value.
`MPI_ERR_LOCKTYPE`	36	Invalid locktype.
`MPI_ERR_NAME`	37	Name not found.
`MPI_ERR_NO_MEM`	38	Memory exhausted.
`MPI_ERR_NOT_SAME`	39	

Symbolic Name	Value	Meaning
MPI_SUCCESS	0	Successful return code.
MPI_ERR_NO_SPACE	40	Not enough space.
MPI_ERR_NO_SUCH_FILE	41	File (or directory) does not exist.
MPI_ERR_PORT	42	Invalid port.
MPI_ERR_QUOTA	43	Quota exceeded.
MPI_ERR_READ_ONLY	44	Read-only file system.
MPI_ERR_RMA_CONFLICT	45	Conflicting accesses to window.
MPI_ERR_RMA_SYNC	46	Erroneous RMA synchronization.
MPI_ERR_SERVICE	47	Invalid publish/unpublish.
MPI_ERR_SIZE	48	Invalid size.
MPI_ERR_SPAWN	49	Error spawning.
MPI_ERR_UNSUPPORTED_DATAREP	50	Unsupported datarep passed to `MPI_File_set_view`.
MPI_ERR_UNSUPPORTED_OPERATION	51	Unsupported operation, such as seeking on a file that supports only sequential access.
MPI_ERR_WIN	52	Invalid window.
MPI_ERR_LASTCODE	53	Last error code.
MPI_ERR_SYSRESOURCE	-2	Out of resources

Appendix C MPI Data_type Handles and Their Meanings

Name	C Data Type
MPI_CHAR	signed char
MPI_WCHAR	wchar_t - wide character
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_LONG_LONG_INT	signed long long int
MPI_LONG_LONG	signed long long int
MPI_SIGNED_CHAR	signed char
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_UNSIGNED_LONG_LONG	unsigned long long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_C_COMPLEX	float _Complex
MPI_C_FLOAT_COMPLEX	float _Complex
MPI_C_DOUBLE_COMPLEX	double _Complex
MPI_C_LONG_DOUBLE_COMPLEX	long double _Complex
MPI_C_BOOL	_Bool
MPI_C_LONG_DOUBLE_COMPLEX	long double _Complex
MPI_INT8_T	int8_t
MPI_INT16_T	int16_t
MPI_INT32_T	int32_t
MPI_INT64_T	int64_t
MPI_UINT8_T	uint8_t
MPI_UINT16_T	uint16_t
MPI_UINT32_T	uint32_t
MPI_UINT64_T	uint64_t
MPI_BYTE	8 binary digits
MPI_PACKED	data packed or unpacked with MPI_Pack()/ MPI_Unpack

Appendix D Background Mathematics

D.1 A Bit About Graphs

A ***graph*** G consists of two finite sets, V and E. Each element of V is called a ***vertex***, the plural of which is ***vertices***. The elements of E are called ***edges***, and they consist of *unordered pairs* of vertices from the vertex set V. Formally, $E \subseteq V \times V$. Graphs can be depicted visually by creating a node for each vertex and a line segment for each edge. If (s,t) is an edge we would draw a line from s to t. If the pairs of edges are ordered pairs, then the graph is called a ***directed graph***, and we draw the lines between nodes with arrows on their ends.

Example 1. Let $G =< V, E >$be a directed graph with the vertex set $V = \{a, b, c, d, e\}$, and the edge set $E = \{(a, b), (a, c), (b, d), (c, d), (d, e), (e, a)\}$. Then we would draw this graph as shown in Figure D.1.

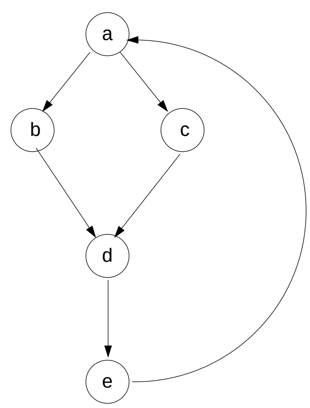

Figure D.1: A directed graph with 5 vertices.

Notice that there is a path, meaning a sequence of edges, from the first vertex a through either b or c then through d, then e and back to a. This is an example of a ***cycle***. When a directed graph has a cycle it is called a ***cyclic graph***. If it has no cycles it is called an ***acyclic graph***.

D.2 Differential Equations

A differential equation is an equation that relates the values of a function and one or more of its derivatives. The function in the equation might be a function of one variable or of several variables. When it is of one variable, it is an ***ordinary differential equation***. When the function has several variables, and the equation contains partial derivatives with respect to these, the equation is called a ***partial differential equation***. For instance

$$f'(x) = 2f(x)$$

is an ordinary differential equation. Can you think of a function whose derivative is always double the value of the function at any point? The exponential function $f(x) = e^{\{2x\}}$satisfies this equation, so $e^{\{2x\}}$is called a solution of it.

It is customary to write these equation in a standard form:

$$f'(x) - 2f(x) = 0$$

An example of a ***second order equation*** is

$$f''(x) + f(x) = 0$$

The *sin*() function is a solution to this equation, since $\frac{d}{dx}(\sin x) = \cos x$ and $\frac{d}{dx}(\cos x) = -\sin x$, so $\frac{d^2}{dx^2}(\sin x) = -\sin x$.

An example of a very well-known partial differential equation is the ***Laplace equation***:

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

This equation arises in many areas of science.

D.2.1 Finite Difference Equations

The approximation of derivatives by finite differences is one very important method of numerically solving differential equations, such as those arising in boundary value problems.

From calculus we know that the first derivative of a function f at a point x is defined by

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{D.1}$$

The first derivative of f at a point x can be approximated by the ***difference quotient***

$$\Delta_h[f](x) = \frac{f(x+h) - f(x)}{h} \tag{D.2}$$

where h is a small constant. This is basically the slope of the line between $f(x)$and $f(x+h)$. The numerator of the difference quotient in Eq. D.2 is called a ***forward difference*** because it is the difference between the value of the function of the point after x and its value at x. We can also approximate the first derivative with a ***central difference***:

$$\delta_h[f](x) = \frac{f(x+h/2) - f(x-h/2)}{h} \tag{D.3}$$

again with h a small constant. Since the second derivative is defined as

$$f''(x) = \lim_{h \to 0} \frac{f'(x+h) - f'(x)}{h}$$

we can approximate it using the central difference as follows:

$$\begin{aligned}
f''(x) &= \frac{f'(x+h/2) - f'(x-h/2)}{h} \\
&= \frac{\frac{f(x+h/2+h/2) - f(x+h/2-h/2)}{h} - \frac{f(x-h/2+h/2) - f(x-h/2-h/2)}{h}}{h} \\
&= \frac{\frac{f(x+h) - f(x)}{h} - \frac{f(x) - f(x-h)}{h}}{h} \\
&= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \qquad \text{(D.4)}
\end{aligned}$$

D.2.2 Example

As an example of the application of this finite difference method, in Chapter 3, we saw that the one-dimensional heat equation was solved using the equation

$$u_{i,j+1} = u_{i,j} + \frac{k \cdot u_{i-1,j} - 2k \cdot u_{i,j} + k \cdot u_{i+1,j}}{h^2}$$

where the value $u_{i,j}$ represented the heat at position i on a thin rod at time j and h was the distance between successive points on the rod, and k was the difference between successive time intervals, so that $u_{i+1,j} - u_{i,j} = h$ and $u_{i,j+1} - u_{i,j} = k$. The heat transfer equation in one dimension is

$$\frac{\partial u}{\partial t} - \alpha \frac{\partial^2 u}{\partial x^2} = 0$$

where α is a constant that we can assume is 1. Using the above finite difference formulas, this equation becomes

$$\frac{u(x,t+k) - u(x,t)}{k} = \frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{h^2}$$

because u is a function of both time and position. If we replace $u(x,t+k)$ by $u_{i,j+1}$ and $u(x+h,t)$ by $u_{i+1,k}$ and so on, this formula becomes

$$\frac{u_{i,j+1} - u_{i,j}}{k} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}$$

or

$$u_{i,j+1} - u_{i,j} = \frac{k \cdot u_{i+1,j} - 2k \cdot u_{i,j} + k \cdot u_{i-1,j}}{h^2}$$

and finally

$$u_{i,j+1} = u_{i,j} + \frac{k \cdot u_{i-1,j} - 2k \cdot u_{i,j} + k \cdot u_{i+1,j}}{h^2}$$