MPI — Message Passing Interface

- `MPI_Abort`: Shuts down all processes and returns the specified integer error code. Not very graceful but works.
  `MPI_Abort(MPI_COMM_WORLD, 1);`
  Example: Abort
- `MPI_Scan` and `MPI_Exscan`: Performs a reduction, but it keeps the partial results on the sequential processors. Like a reduce, but leaves the first i elements combined, on processor i. (VE-V2-58)
  Example: Scan
- `MPI_Scatterv` and `MPI_Gatherv`: Vector forms for the scatter and gather routines. (VE-V2-75)
  Example: ScatterGatherV
  Example: ScatterGatherV2
- `MPI_Allgatherv`: Vector form for the allgather routine. (VE-V2-76)
  Example: ScatterAllGatherV

MPI — Message Passing Interface

- `MPI_Alltoall`: Similar to a collection of simultaneous broadcasts or simultaneous gathers. This is more of a transposing of data between processors. It takes the first *m* items from process 0 to process 0, then then next *m* from process 0 to process 1, the next *m* to process 2, and so on. Once the data from process 0 is finished it takes the next *m* from process 1 to process 0, then 1 to 1, 1 to 2, and so on. (VE-V2-67)
  Example: AllToAll
- `MPI_Probe`: Command to request the message status before receiving the message. Can be used to save memory with the receiving buffer and to keep from a receiving buffer being too small. (VE-V2-136)
  Example: Probe
- `MPI_Sendrcv`: Pairwise exchange of data. The sendrecv call works great if every process is paired with precisely one sender and one receiver. (VE-V2-111)
  Example: SendRecv

MPI — Message Passing Interface

- `MPI_Sendrecv_replace`: Like the `MPI_Sendrcv` except that the buffer is both input and output. (VE-V2-115)
  Example: SendRecv2
- `MPI_Allreduce(MPI_IN_PLACE, ...`: Overwrites the reduction variable with the reduction result on all processors. Hence there is no need for duplicate memory space, can make a difference when reducing large arrays. (VE-V2-51)
  Example: AllReduceInPlace

Reference Roadmap

The slide outlines contain references to the main course materials. Not everything has a reference but nearly all the materials can be found in the following references. The references are of the form `(<text>-<pages>)` so (PM-123) means page 123 of An Introduction to Parallel Programming by Peter S. Pacheco and Matthew Malensek.

- (VE-V1) — The Science of Computing: The Art of High Performance Computing, Vol 1 by Victor Eijkhout
- (VE-V2) — Parallel Programming in MPI and OpenMP: The Art of HPC, Vol 2 by Victor Eijkhout
- (VE-V3) — Introduction to Scientific Programming in C++17/Fortran2008: The Art of HPC, Vol 3 by Victor Eijkhout
- (VE-V4) — Tutorials for High Performance Scientific Computing: The Art of HPC, Vol 4 by Victor Eijkhout
- (PM) — An Introduction to Parallel Programming by Peter S. Pacheco and Matthew Malensek.
- (KH) — Programming Massively Parallel Processors: A Hands-on Approach by David B. Kirk and Wen-mei W. Hwu.
- (SAB) — High Performance Computing Modern Systems and Practices by Thomas Sterling, Matthew Anderson, and Maciej Brodowicz.
- (GLS) — Using MPI: Portable Parallel Programming with the Message-Passing Interface by William Gropp, Ewing Lusk, and Anthony Skjellum.
- (GHTL) — Using Advanced MPI: Modern Features of the Message-Passing Interface by William Gropp, Torsten Hoefler, Rajeev Thakur, and Ewing Lusk.