# MPI: 25 Years of Progress

Anthony Skjellum

University of Tennessee at Chattanooga

Tony-skjellum@utc.edu

Formerly: LLNL, MSU, MPI Software Technology,
Verari/Verarisoft, UAB, and Auburn University

Co-authors: Ron Brightwell, Sandia
Rossen Dimitrov, Intralinks

# MPI: 25 Years of Progress

Anthony Skjellum

University of Tennessee at Chattanooga

Tony-skjellum@utc.edu

Formerly: LLNL, MSU, MPI Software Technology, Verari/Verarisoft, UAB, and Auburn University

Co-authors: Ron Brightwell, Sandia
Rossen Dimitrov, Intralinks

# Outline

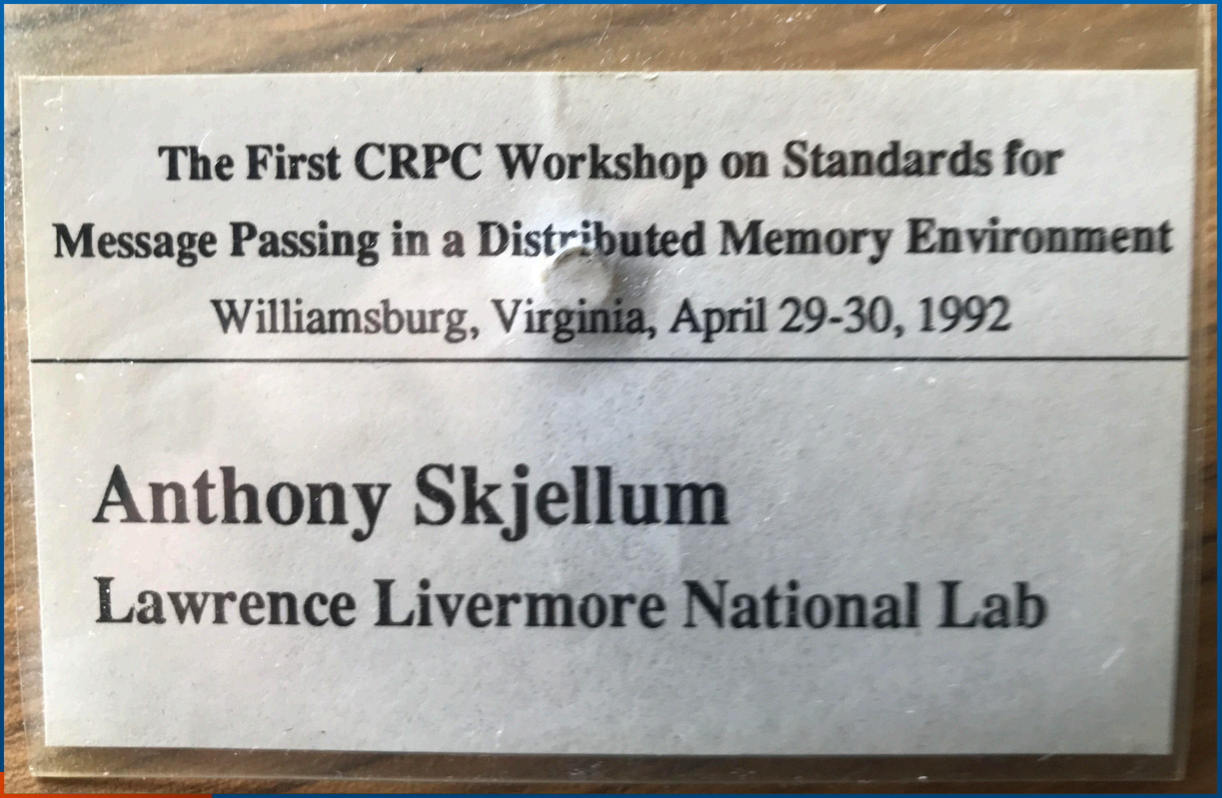
- Background
- Legacy
- About Progress
- MPI Taxonomy
- A glimpse at the past
- A look toward the future

# Progress

- 25 years we as a community set out to standardize parallel programming
- It worked ☺
- Amazing “collective operation” (hmm.. still not complete)
- Some things about the other progress too, moving data independently of user calls to MPI…

# Community

- This was close to the beginning…



The First CRPC Workshop on Standards for

Message Passing in a Distributed Memory Environment

Williamsburg, Virginia, April 29-30, 1992

Anthony Skjellum

Lawrence Livermore National Lab

# As we all know (agree?)

- MPI defined progress as a “weak” requirement
- MPI implementations don’t have to move the data independently of when MPI is called
- Implementations can do so
- There is no need for an internally concurrent schedule to comply
- For instance: do all the data movement at “Waitall” ··· predictable if required only to be here!

# How programs/programmers achieve progress

- The MPI library calls the progress engine when you call any of most MPI calls
- The MPI library does it for you
  - In the transport, MPI just shepherds lightly
  - In an internal thread or threads periodically scheduled
- You kick the progress engine (Self help)
  - You call MPI_Test() sporadically in your user thread
  - You schedule and call MPI_Test() in a helper thread

# Desirements

- Overlap communication and Computation
- Predictability / low jitter

- Later: overlap of communication, computation, and I/O

- Proviso: LJ → Must have the memory bandwidth

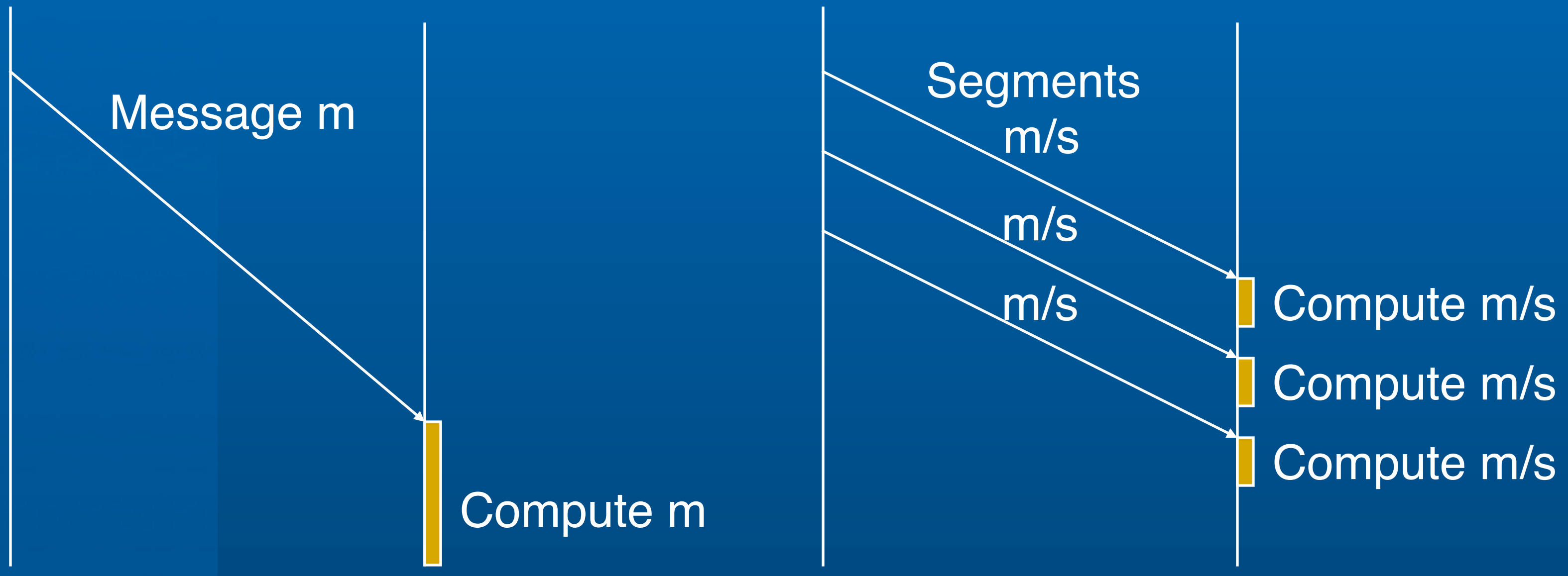
# MPI Implementation Taxonomy (Dimitrov)

- Message completion notification
  - Asynchronous (blocking)
  - Synchronous (polling)
- Message progress
  - Asynchronous (independent)
  - Synchronous (polling)

| | |
|---|---|
| blocking independent | blocking polling |
| polling independent | all-polling |

# Segmentation

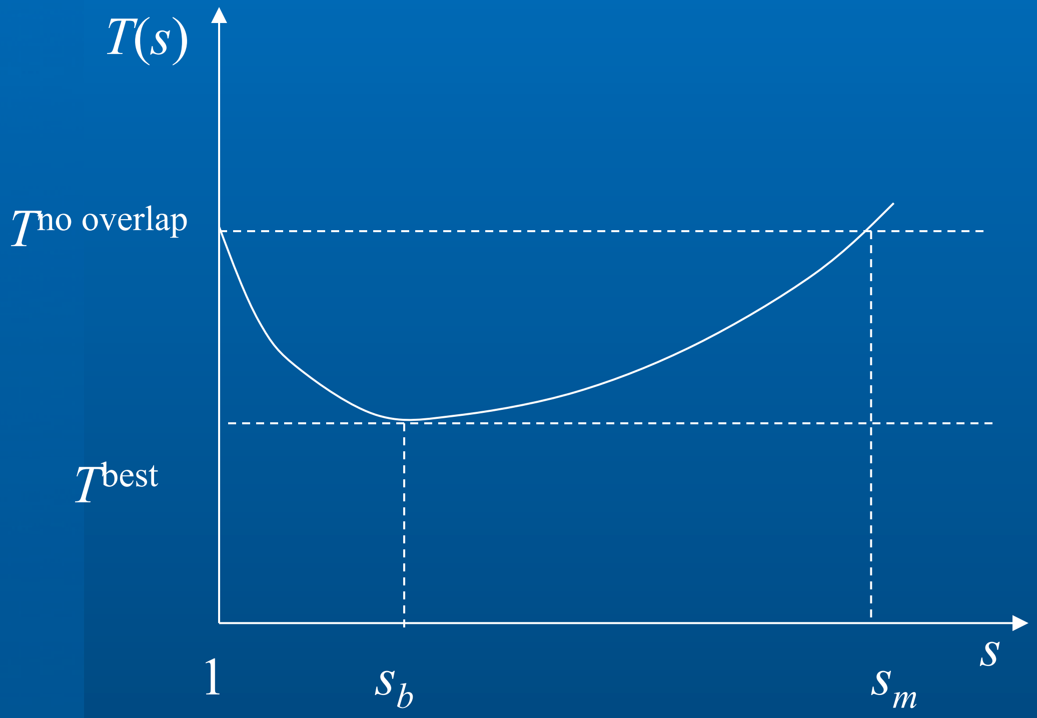
- Common technique for implementing overlapping through pipelining



Entire message

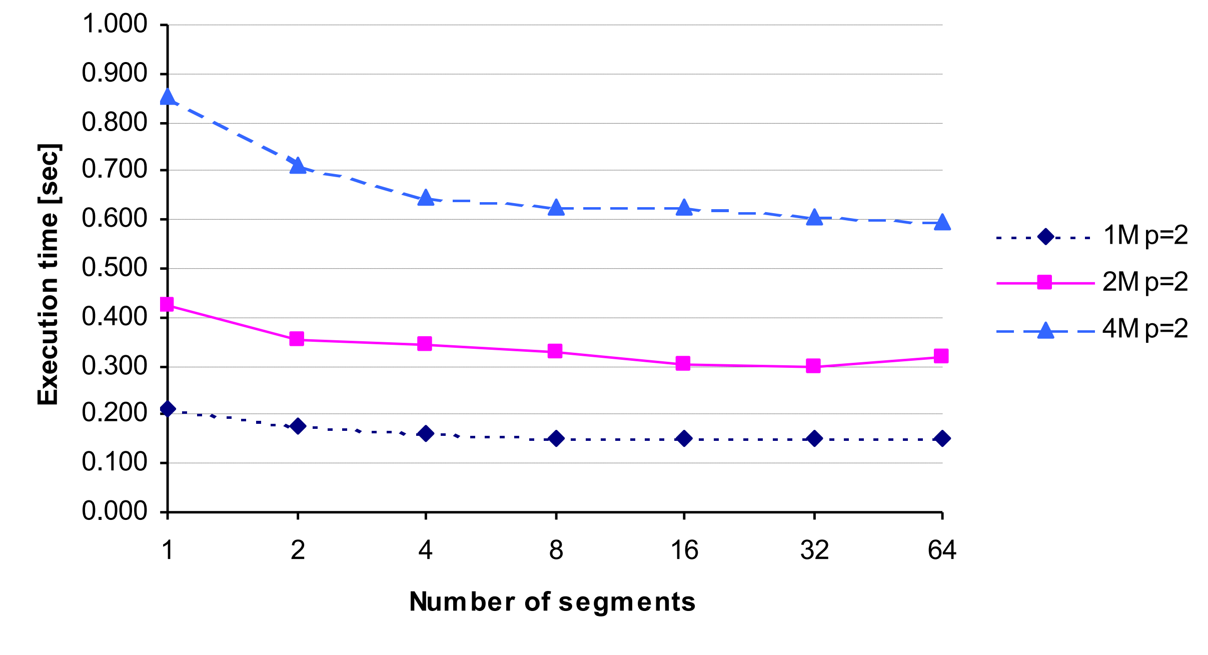
Segmented message

# Optimal Segmentation



# Performance Gain from Overlapping

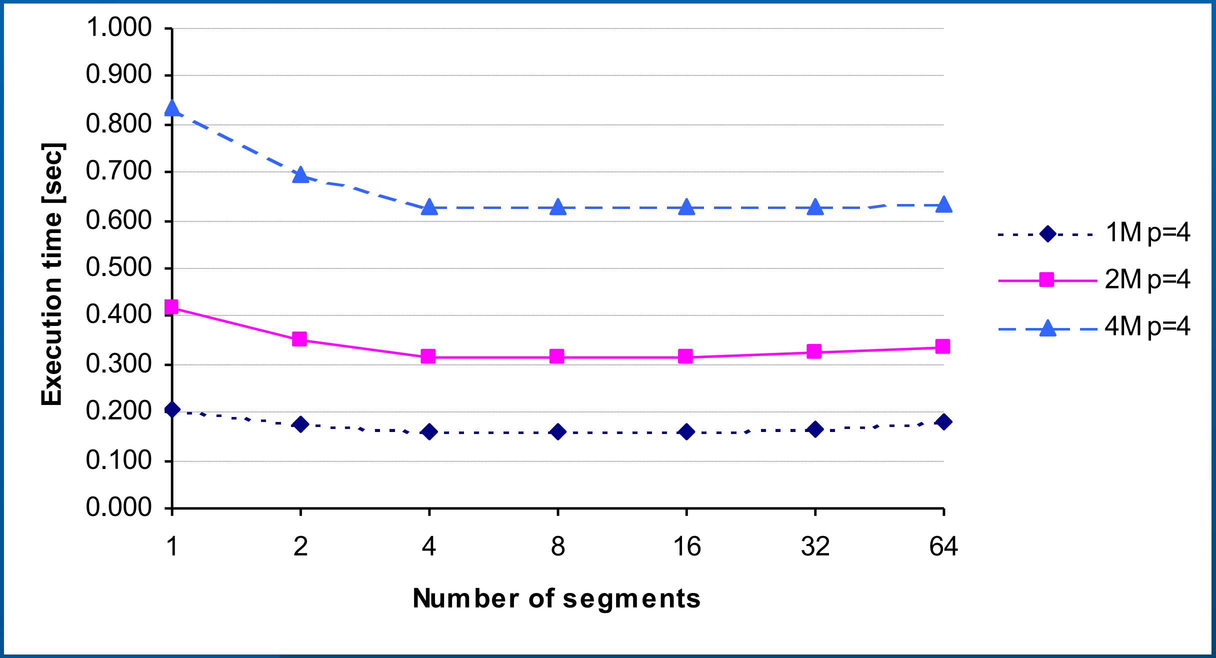
- Effect of overlapping on FFT global phase in seconds, p = 2



| size | Max speedup |
|---|---|
| 1M | 1.41 |
| 2M | 1.43 |
| 4M | 1.43 |

# Performance Gain from Overlapping (cont.)

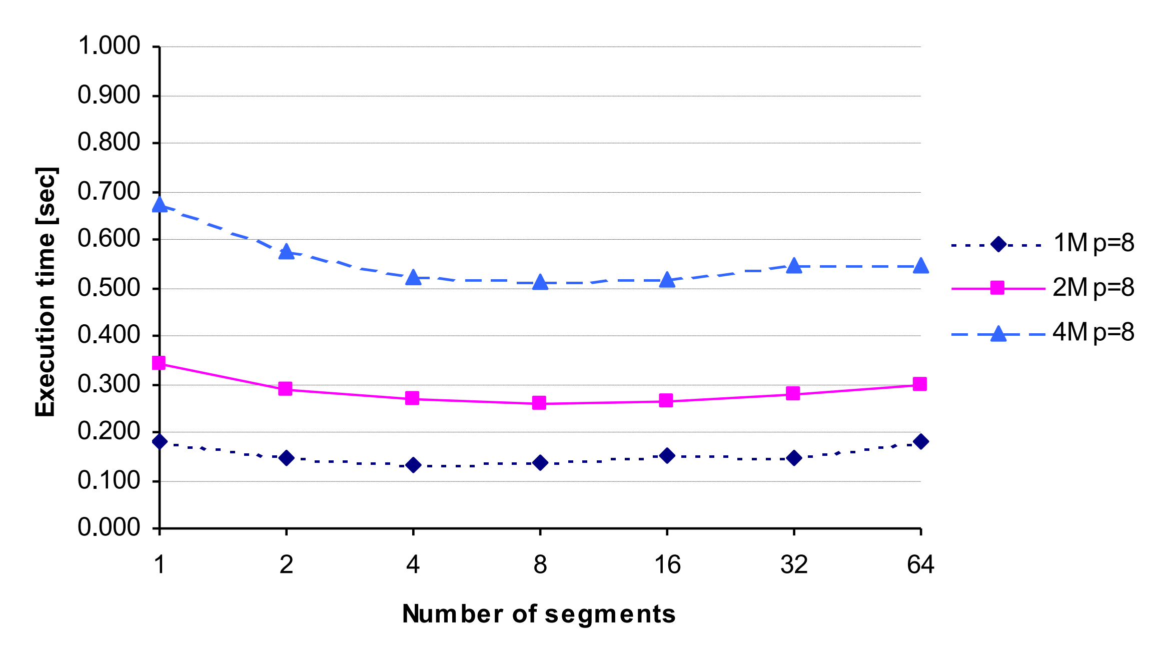
- Effect of overlapping on FFT global phase in seconds, p = 4



| size | Max speedup |
|---|---|
| 1M | 1.31 |
| 2M | 1.32 |
| 4M | 1.33 |

# Performance Gain from Overlapping (cont.)

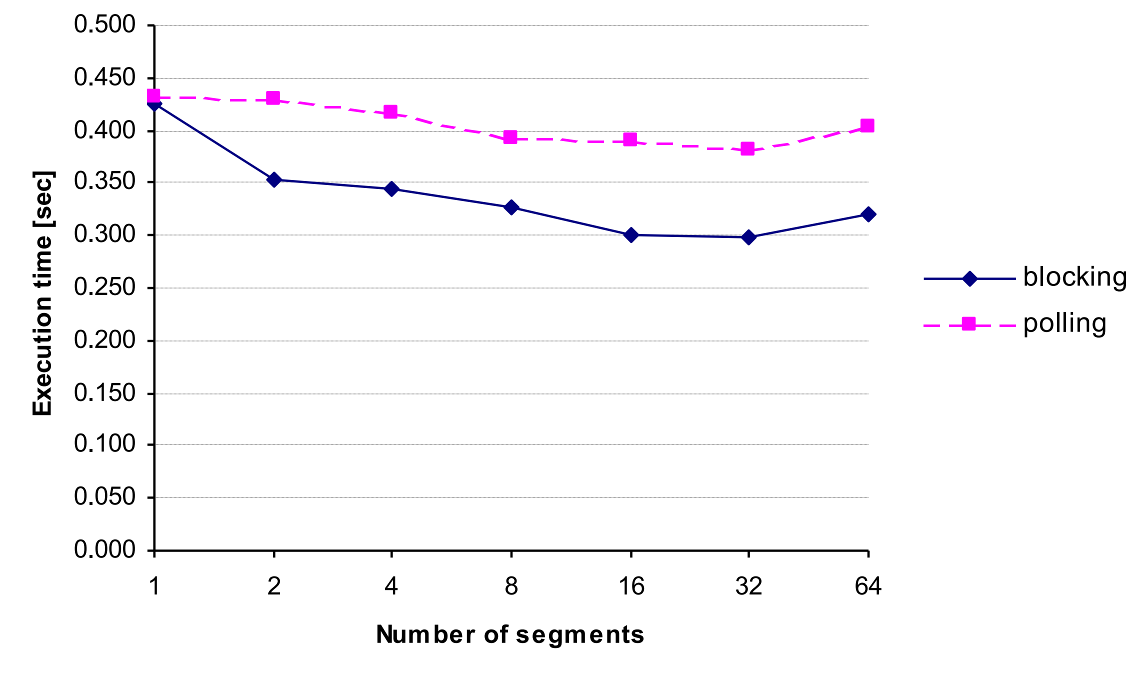
- Effect of overlapping on FFT global phase in seconds, p = 8



| size | Max speedup |
|---|---|
| 1M | 1.32 |
| 2M | 1.32 |
| 4M | 1.33 |

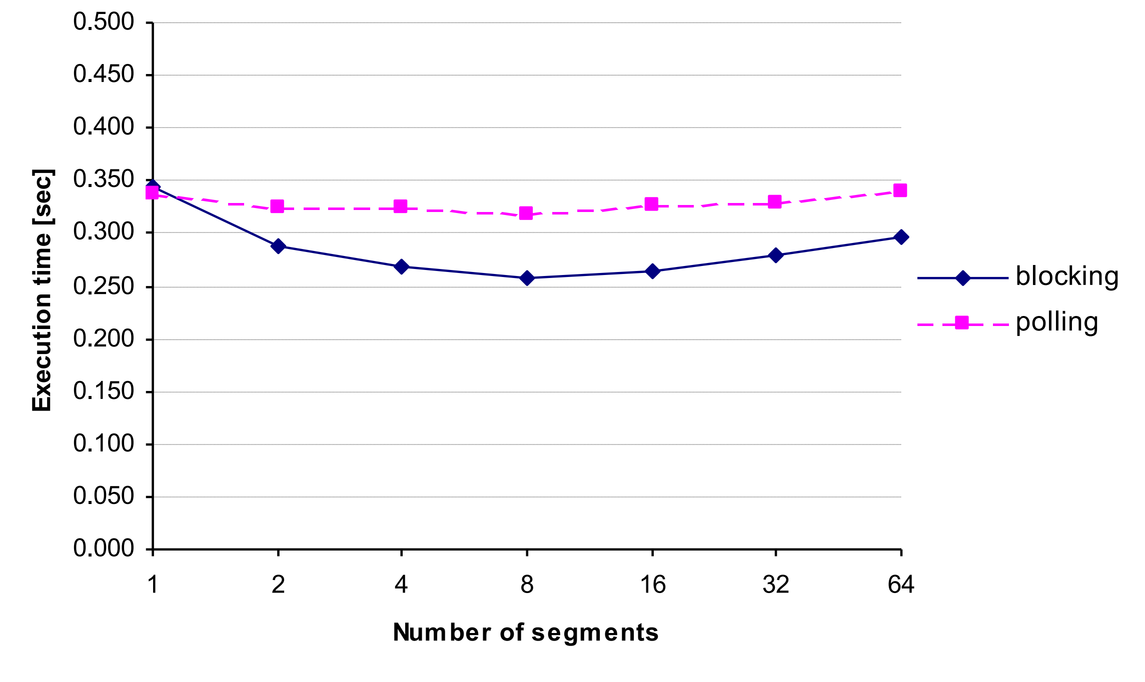
# Effect of Message-Passing Library on Overlapping

- Comparison between blocking and polling modes of MPI, n = 2M, p = 2



# Effect of Message-Passing Library on Overlapping

- Comparison between blocking and polling modes of MPI, n = 2M, p = 8



# Observations/Upshots

- Completion notification method affects latency of short messages (i.e., < 4k on legacy system)
- Notification method did not affect bandwidth of long messages
- Short message programs
  - Strong progress, polling notification
- Long message programs
  - Strong progress, blocking notification

# Future (soon?)

- MPI's support overlap and notification mode well
- Overlap is worth at most a factor of 2 (3 if you include I/O)
- It is valuable in real algorithmic situations
- Arguably growing in value at exascale
- We need to reveal this capability broadly without the "Self help" model

# Thank you

- 25 years of progress
- And still going strong···
- Collective!
- Nonblocking?
- Persistent!
- Fault Tolerant?