Large Counts in MPI

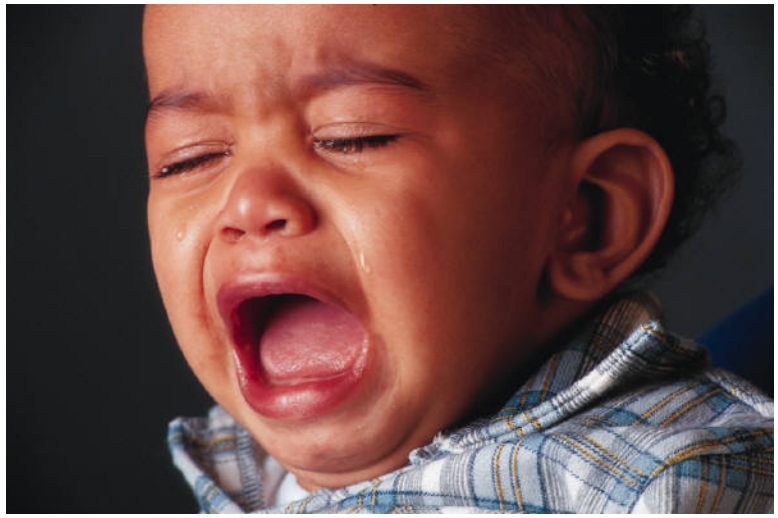
A Brief History

How it all started

- HP-MPI users request 64-bit counts (nat'l labs)
- HP takes issue up in Forum, Ticket #14, September 2008
- Proposes new extended functions for inclusion in MPI 2.2 (MPI_SendL, etc)
- Covers pt2pt, collective, datatypes, rma
- Uses MPI_Aint for count type

Forum Feedback

- Too big a change for 2.2, delay to 3.0
- Define new MPI_Count type instead of using MPI_Aint



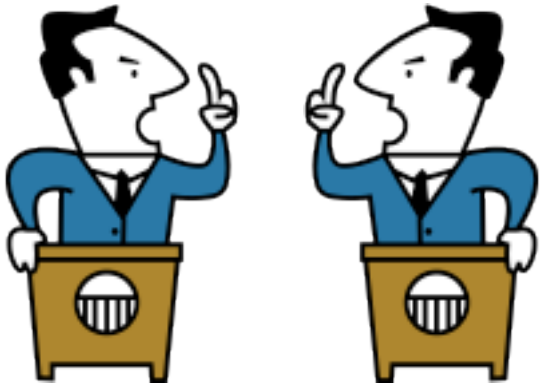
- Leads to ticket #117, Feb 2009

Ticket #117

- Modifies existing APIs
- Covers pt2pt, collective, datatypes, rma
- Uses MPI_Count for count type

Forum Feedback

- Lots of thrashing
  - Does anyone really *need* MPI_Count?
  - Does new functionality use MPI_Count?
  - What's the impact on Fortran?
  - Is the MPI_Count version of all APIs too big of an API explosion?
- HP sells HP-MPI, ticket lays idle for a while



Ticket #220

- Focus shifts from communication to MPI-IO
- Ticket #117 resurrected, changed in scope to just:
  - MPI_GET_COUNT_L
  - MPI_GET_ELEMENTS_L
  - MPI_STATUS_SET_ELEMENTS_L
- Ticket #220 introduces _L version of MPI-IO routines.
- Back to _L suffix!

Forum Feedback

- Discussion about suffixing rules
  - What happens when the next suffix is introduced?
    - i.e. timeout parameter
- Generally rejected by forum as being too big of a change for too little benefit

- Dave Gives up.

Current Proposal

- Define MPI_Count as >= int
  - Define as int for back compatibility
- Apply it to all APIs
- No suffixing
- Requires two libs
  - back compat
  - long counts
- Can be handled in single codebase
- Implementers decide which to ship/support