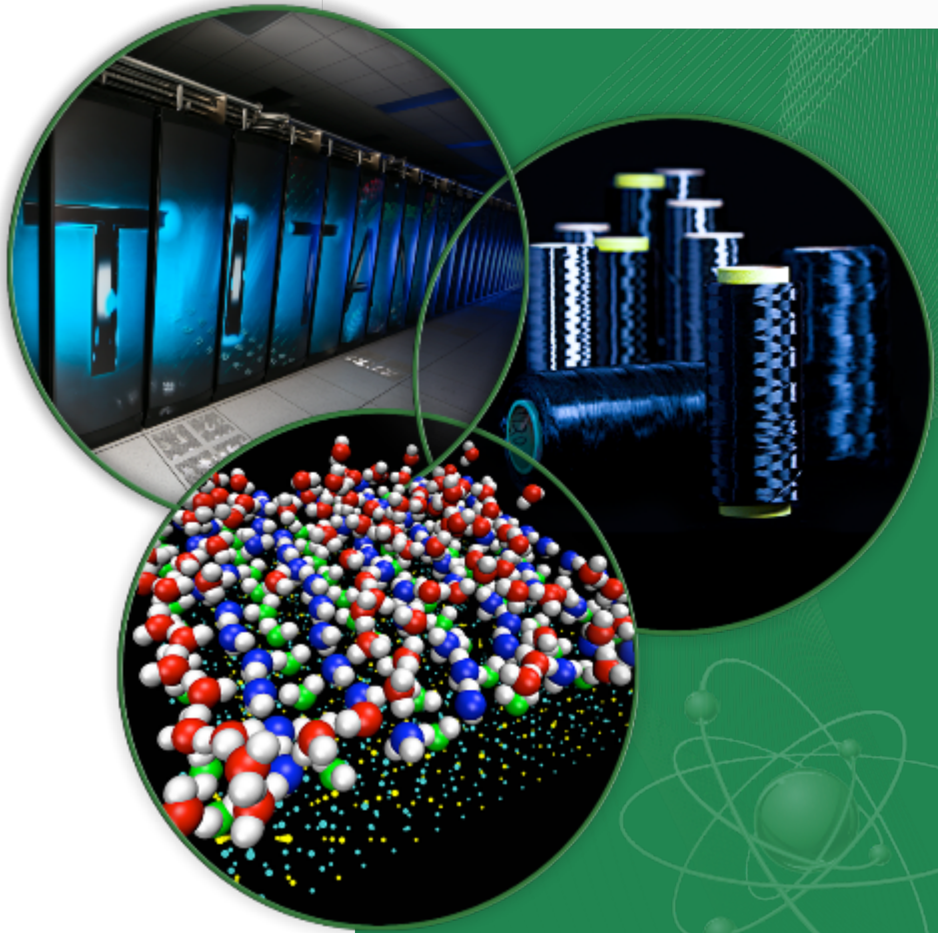
Birth of a De Facto Standard Message Passing Interface

Al Geist

ORNL

Celebrating 25 years of MPI

September 25, 2017
ANL



Birth of a De Facto Standard Or How I Stopped Worrying and Learned to Hate Dallas

- In 1992 Jack, Rolf, and Tony hold a meeting to try to get vendors to adopt a single message passing standard. Some vendors want it to be **PVM**, but other vendors want it to be their personal API.
  - At this meeting it became clear that no existing API would be adopted. So the HPC community would have to collectively create a message passing interface that everyone could feel ownership of.

- 1993 MPI 1.0 Forum meets in Dallas every 6 weeks for most of the year to create the MPI 1.0 API

Remember getting bussed to the hotel?

- 1995 MPI 2.0 Forum meets in Chicago airport every 6 weeks for 2 years to create the MPI 2.0 API

Just walked across street at O'Hare

Couple Mid-wives help with the Birth

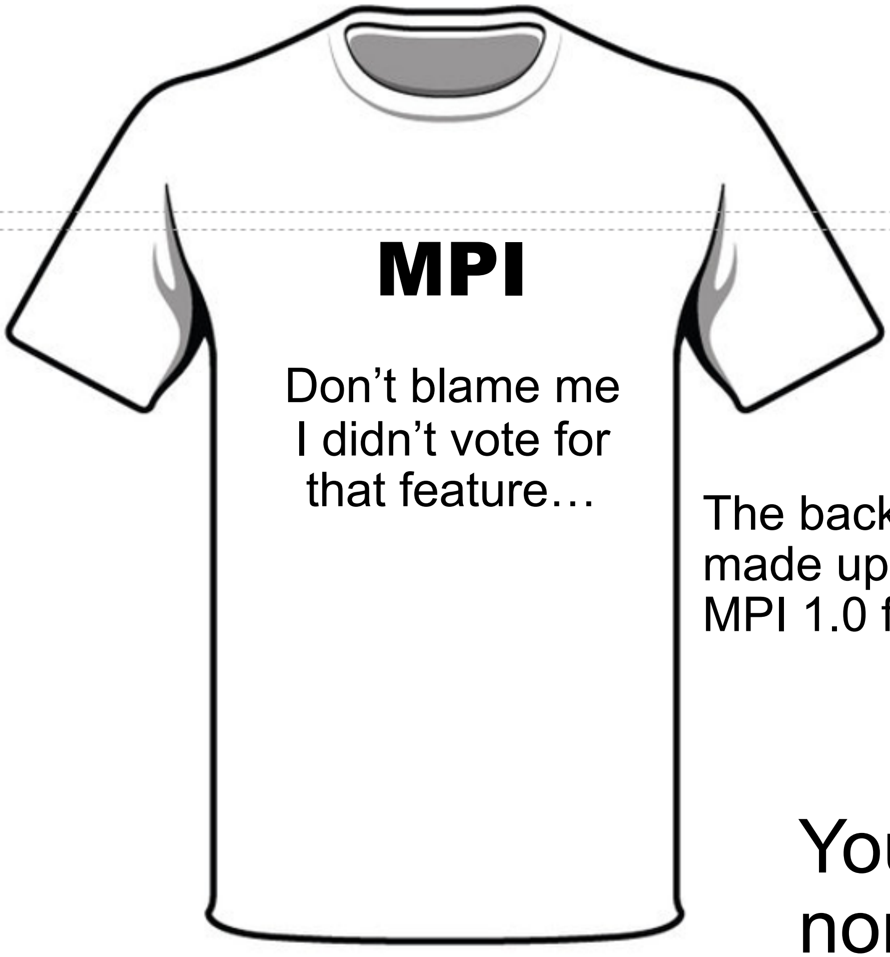
MPI use grew, but didn't become a de facto standard till around 2000 when its user base finally grew larger than PVM

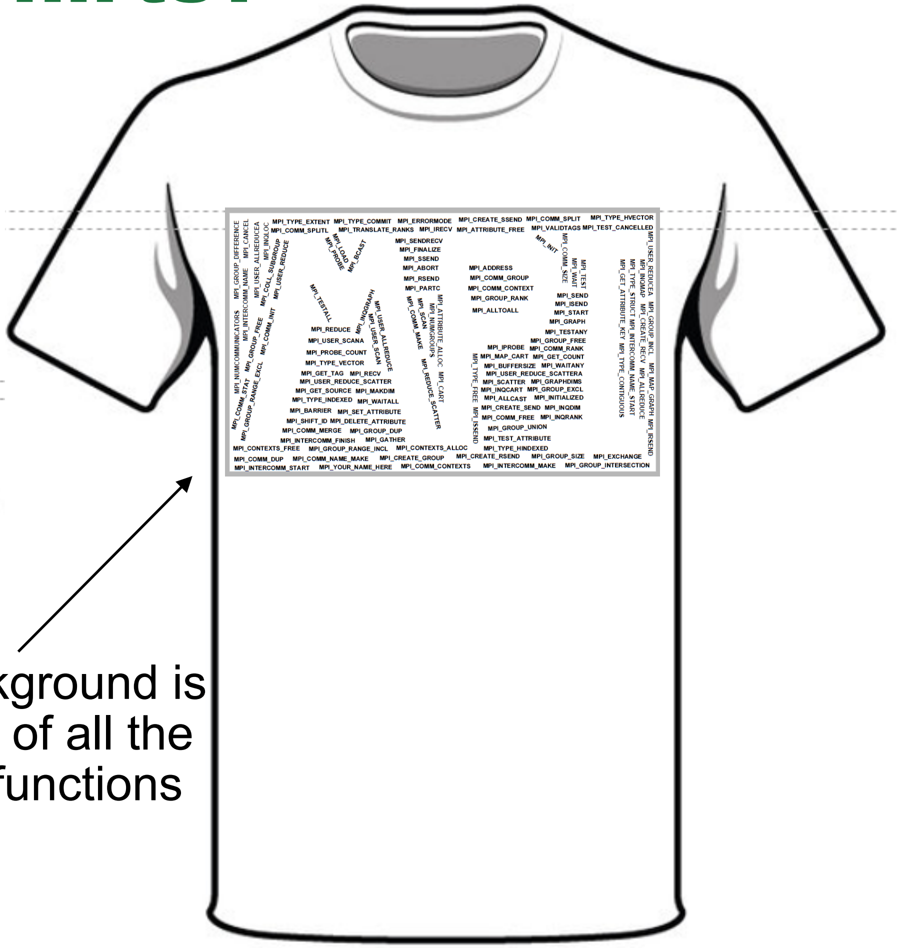
Dan Hitchcock (DOE CS program manager) and his boss Walt Polanski. Dan called me in 1998 at the peak of PVM use and said he was canceling all funding for PVM research – go do something else.

And so we did, which helped MPI adoption and the establishment of a single de facto standard

EuroPVM **EuroPVM/MPI** **EuroMPI**

Remember the MPI Shirts?





The background is made up of all the MPI 1.0 functions

You want a non-blocking what???…

The MPI Non-Blocking Barrier

- While folks in this room know why you want this function. The general user always looks at me like "The MPI Forum must be crazy"
- The t-shirts reflect some of the feedback the community felt early on about MPI
  - The Monet t-shirt: Reflected the thing we often heard "MPI has way too many functions"
  - Don't blame me. I didn't vote for that feature. . . Reflected the many new concepts introduced by MPI that one has to use.
  - You want a non-blocking what? Reflected that we made sure there was a non-blocking version of everything – even the blocking function

When MPI is Your Hammer Every Problem Looks like a Thumb

Marc Snir's talk covers this:
"MPI is too High-level;
MPI is too Low-level"

MPI give the users many ways to tackle their problems.

Leave it to our creative users to find poor ways to use the MPI functions.

Failing to Define a Fault Tolerant MPI

A Regret:

That we were unable to define MPI to allow applications to "run through" faults rather than abort the entire parallel job when one node fails.

Championed by Al Geist in MPI 1.0 and MPI 2.0 and by Rich Graham in MPI 3.0.

For 25 years the MPI Forum has always voted this capability down

It was a common complaint by users,
but now they seem resigned to MPI's behavior

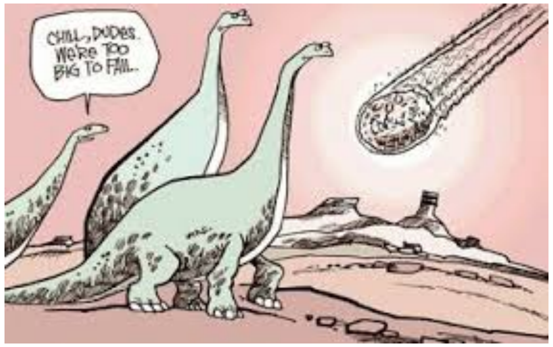
It was possible as demonstrated by UTK's FT-MPI Research (and others)

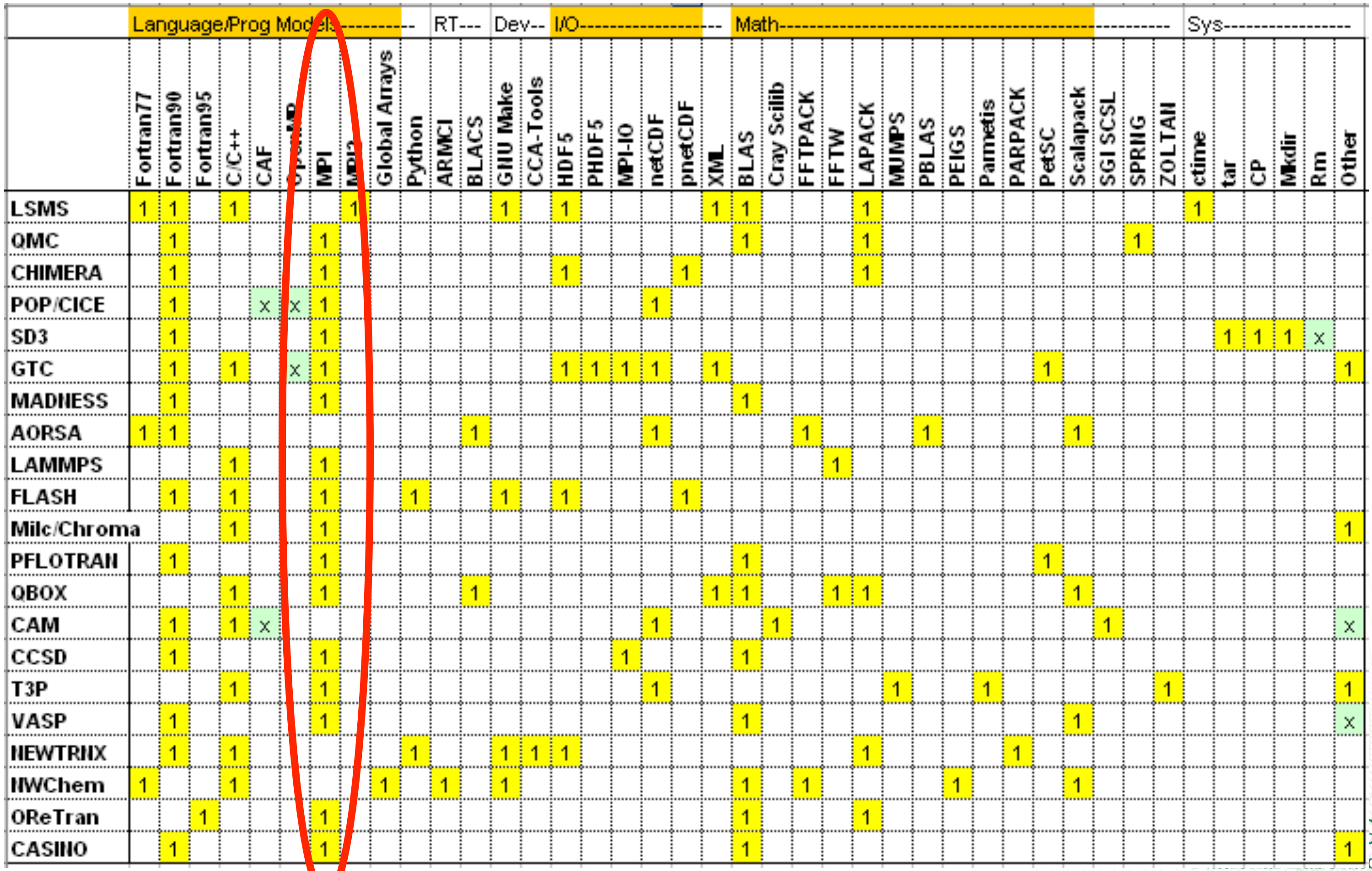
- Define the behavior of MPI in case an error occurs
- Give the application the possibility to recover from a node-failure
- Provide the notification to the application
- Provide recovery options for the application to exploit if desired

Aborting entire MPI job is a real problem given the resilience of existing systems (Seen this year on Titan)

MPI Too Big To Fail

MPI is the dominant programming method used in todays HPC science apps





	Language/Prog Models----------										RT---		Dev--		I/O------------------						Math--															Sys----------------					
	Fortran77	Fortran90	Fortran95	C/C++	CAF	OpenMP	MPI	MPI2	Global Arrays	Python	ARMCI	BLACS	GNU Make	CCA-Tools	HDF5	PHDF5	MPI-IO	netCDF	pnetCDF	XML	BLAS	Cray Scilib	FFTPACK	FFTW	LAPACK	MUMPS	PBLAS	PEIGS	Parmetis	PARPACK	PetSC	Scalapack	SGI SCSL	SPRNG	ZOLTAN	ctime	tar	cp	Mkdir	Rm	Other
LSMS	1	1		1				1					1		1					1	1				1											1					
QMC		1					1														1				1									1							
CHIMERA		1					1								1				1						1																
POP/CICE		1			x	x	1											1																							
SD3		1					1																														1	1	1	x	
GTC		1		1		x	1								1	1	1	1		1											1										1
MADNESS		1					1														1																				
AORSA	1	1										1						1					1				1					1									
LAMMPS				1			1																	1																	
FLASH		1		1			1			1			1		1				1																						
Milc/Chroma				1			1																																		1
PFLOTRAN		1					1														1										1										
QBOX				1			1					1								1	1			1	1							1									
CAM		1		1	x													1				1											1								x
CCSD		1					1										1				1																				
T3P				1			1											1								1			1						1						1
VASP		1					1														1											1									x
NEWTRNX		1		1						1			1	1	1										1					1											
NWChem	1			1					1		1		1								1		1					1				1									
OReTran			1				1														1				1																
CASINO		1					1														1																				1

The Answer is MPI.
What is the Question?

Will MPI still be used at Petascale? What about at Exascale?

Applications will continue to use MPI due to:

- Inertia – these codes take decades to create and validate
- Nothing better – developers need a BIG incentive to rewrite (not 50%)

Communication libraries are being changed to exploit new HPC systems, giving applications more life.

- Hardware support for MPI is pushing this out even further

Can MPI Scale to Exascale?

It was a serious topic in 2009 when we tried to launch the Exascale program

MPI can Scale to Exascale

It was a serious topic in 2009 when we tried to launch the Exascale program

Extreme-scale Simulator (X-Sim) developed by Christian Engelmann at ORNL to answer this question

- In 2010 simulated an MPI app running on 1 million processors
- In 2011 simulated an MPI app running on 100 million processors

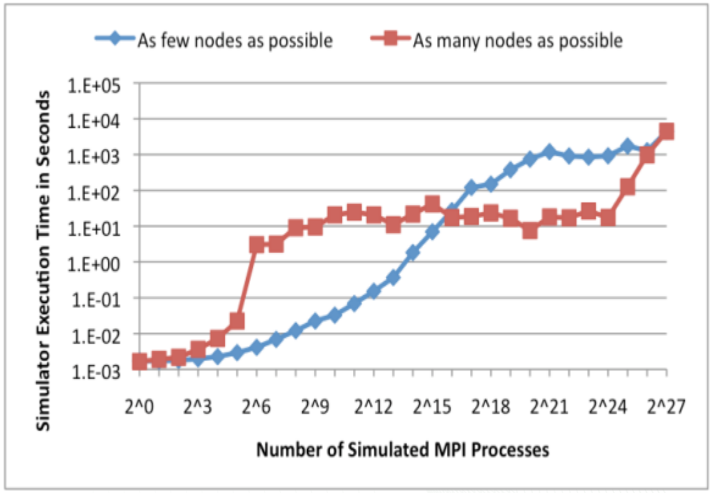
Simulator is a parallel application

– Runs on a Linux Cluster

Adjustable Topology

– Configured at startup

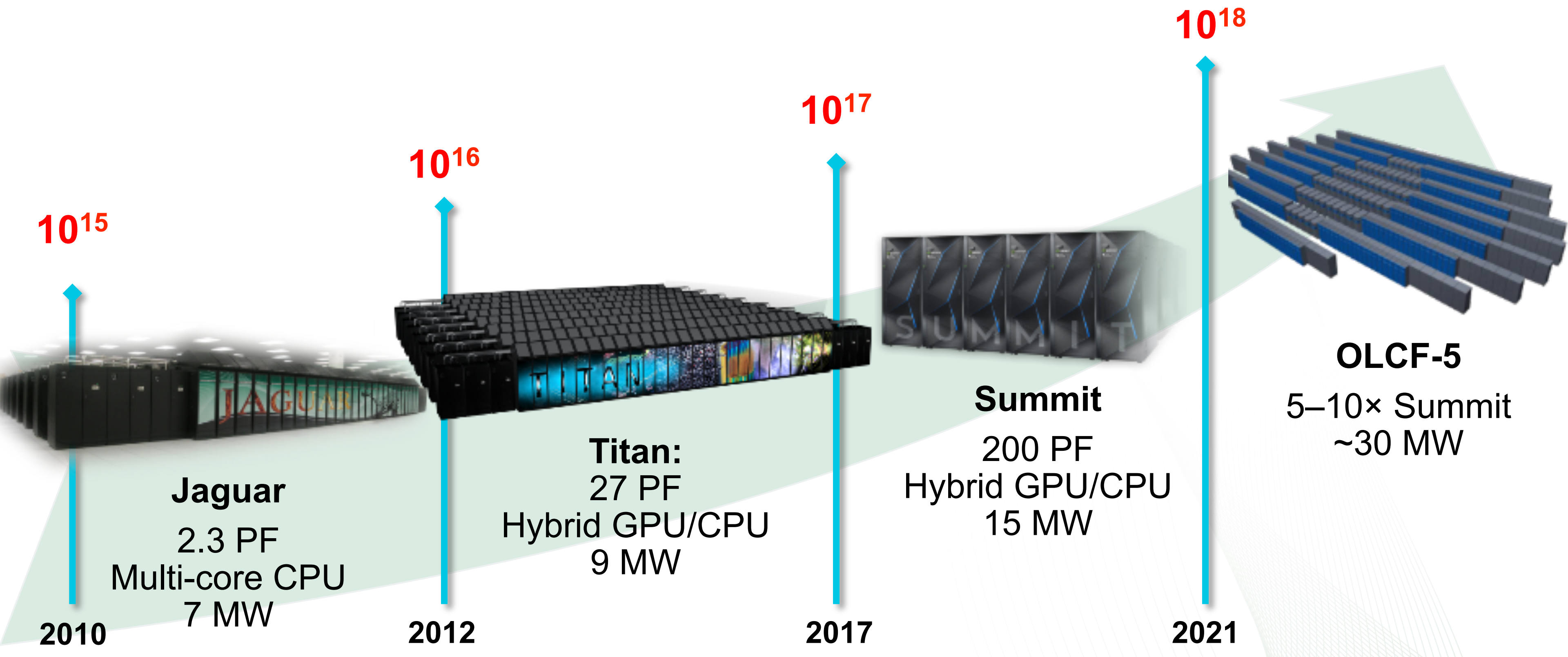
Supports Fortran, and C applications



MPI app Scaling to 134,217,728 (2^27) simulated MPI ranks

MPI will be with us as we march to Exascale

But will it be MPI 4.0 by then???
Yes! . . . according to Steve's strait line graph



Thanks