MPI @ 35

Dan Holmes

EuroMPI 2017

25th Anniversary Symposium

Could you please predict something for me?

(Fore)knowledge *vs.* prediction

"Those who have knowledge, don't predict. Those who predict, don't have knowledge."

Lao Tzu, *6th Century BC Chinese Poet*

We already know

what the future of MPI

will look like

MPI @ 50

Dan Holmes

EuroMPI 2027

35th Anniversary Symposium

Follow me,
I know the way!

Regulation “is the frozen memory of past disaster”

@GeorgeMonbiot, quoting Steven Poole (Guardian newspaper)

“Much red tape is the frozen memory of past disaster.

Modern regulatory regimes as a whole
came into being because of
public outrage at the dangerous practices of
unrestrained industry”

Standardisation “is the frozen memory of past success”

Best practice is the frozen memory of past success.

Modern standards came into being because of public outrage at the dangerous* practices of unrestrained innovation.

* dangerous to portability and so to productivity and consequently to publication rate or to profit

Principles for future of MPI

- No change, unless there is a very good reason
  - "If it ain't broke, keep messin' with it 'til it is"

- What things are ~~wrong~~ sub-optimal that need fixing? What is the best fix?
- What things are ~~missing~~ incomplete that need adding? What is the best addition?

- Who cares? Who wants/needs the fix/new feature?

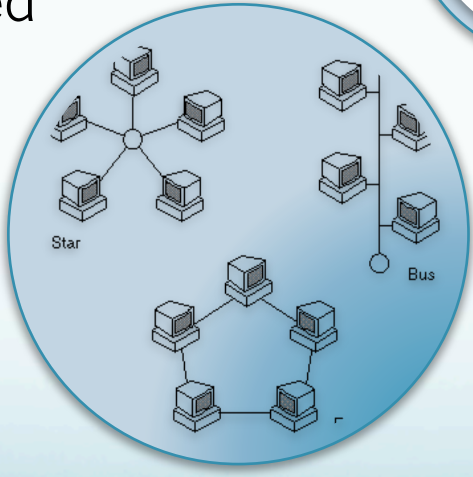
Current MPI: everything an HPC application needs?

Network

- Point-to-point
- Single-sided
- Collective
- I/O

Compute

- Processes
- Groups
- Communicators
- Spawn
- Connect/accept
- Join



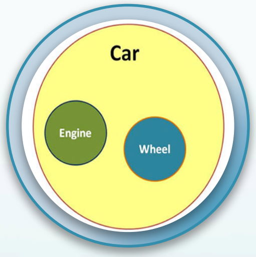
Memory

- Data-types
- Windows
- Files
- Alloc/dealloc

Current MPI: everything an HPC application needs?

Tools
- Profiling
- Debugging
- Adapting

Communication
- Point-to-point
- Single-sided
- Collective
- I/O



Abstractions
- Data-types
- Files
- Windows
- Communicators
- Processes
- Messages

Endpoints/Fine-points

- Process is the fundamental unit of compute in MPI
- Ubiquitously MPI process is mapped to OS process
- What about thread (including SMT & GPU ‘core’)?
  - Who cares? Is hybrid MPI+X actually better?
  - Performance: maybe; abstraction: probably
- Or task (OpenMP, OmpSs, StarPU, PaRSEC)?
  - Who cares? Task-based runtime developers
  - Tasks improve scheduling for less predictable apps
- What is the best fix?

Perfect scenario for MPI

- Regular problem domain
  - Easy to reason about, predict conflicts/bottlenecks
- Predictable hardware
- Long-running, repetitive, iterative algorithm
  - Possible to amortise large setup time/cost
- No surprises
  - Acceptable to dedicate/lock all available resources
- Who cares? Most traditional MPI users

Performance planning

- We have relied on compilers for decades
  - I rarely worry about how many registers my code uses
- MPI middleware augments the compiler
  - Adding distributed-memory communication
- MPI must shoulder some of the burden
  - How many queue-pairs does my code need?
- MPI already does some of this but …
- Expanding the concept of persistence should help

Persistence in MPI

- Currently have persistent point-to-point
  - Half channel, matching still per operation instance
- Working on persistent collectives
  - Full channel, 1-to-n, n-to-1, n-to-n as needed
- Up next: persistent I/O
- Beyond that persistent point-to-point (revisited)
  - Full channel, or stream, matching during initialisation
- Completeness only: persistent single-sided?

Less good scenarios for MPI

- Non-determinism
  - Race-condition (bad?) or asynchronous algorithm
- Irregularity
  - Irregular meshes, AMR, clustering particles
- Unpredictability
  - Data-dependent control-flow, graph algorithms
- Unreliability
  - Hardware may fail, software may fail
  - Hardware performance can vary, e.g. power limits

Faults

- What can go wrong – and be tolerated or fixed?
- Process fail-stop
  - because process is the fundamental unit in MPI?
- Tolerance or resilience?
  - Make user aware of fault & responsible for recovery?
  - Build in redundancy & fail-over to shield the user?
- Who cares? Users of unreliable hardware systems
- What is the best fix? ULFM, FA-MPI, FENIX, CR? All?

Sessions

Sessions

- Problem: initialisation, single controller or race
  - Who cares? Parallel library writers and users
  - What is the best fix? Safe multi-actor initialisation
- Problem: co-location of workflow ensembles is hard
  - Can be done with connect/accept or spawn or join
  - Who cares? Multi-physics, visualisation/steering, tools
  - What is the best fix? Re-vamp process management?
- Problem: adaptation grow/shrink is hard
  - Who cares? Dynamic (unpredictable) applications
  - What is the best fix? Co-location plus ULFM shrink?

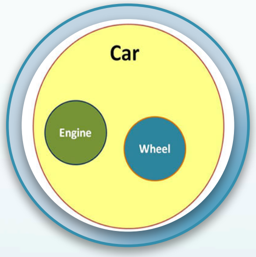
Future MPI: exactly the same, *and* subtly different

Tools

- Profiling
- Debugging
- Visualising
- Steering
- Adapting
- Fault-handling

Communication

- Point-to-point
- Notifications
- Single-sided
- Collective
- Stream
- Event
- I/O



Abstractions

- Data-types
- Files
- Windows
- Communicators
- Endpoints
- Sessions
- Messages

The future is not set. There is no fate but what we make for ourselves.