HPC Languages and Compilation: Où en sommes-nous?

Rob Schreiber

HPC Keynotes, Lyon

29 June 2013

Thanks Alain

- For the conference
- For years of delightful collaboration
- For taking the time to speak to me in French

Welcome to Lyon

Second in population

- First in gastronomy
- Third in soccer
- First in high-level program optimization and hardware synthesis

What is the HPC Language Problem?

- Programmers want a simple model
  - portable
  - durable
  - high performance
- Fortran, C, and MPI
- No pain, no gain?

Let us not forget

- A very brief history of programming in HPC.

The Past (in Brief)

- Vector – we solved it with compilation and a simple programmer’s model
- Scale-out – We learned to live with message passing
    - MPI is a big success (at making MP no worse than necessary)
    - We have tried other models; they work; they have some successes

The Present

- The focus is on “the node,” which has become a **bear** to program
  - Many cores
  - NUMA
  - GPU

Bears

← *To proponents, enthusiasts, Linpack benchmarkers*

To most of the rest of us →

Is Compilation the Solution?

- Dusty decks cannot be used
- Scale out?
- Real codes as they currently exist are usually intractable
  - Parallelization is not a local optimization

The Hardware Future

- More parallel
- Core heterogeneity
- Nonvolatile memory
  - Fast storage
  - Larger capacity
- A diverse collection of acceleration devices
- Photonic networks
  - More node bandwidth and bisection bandwidth
- Resilience. Silent error.

Do we have an intractable problem?

- It is certainly hard

- Why?

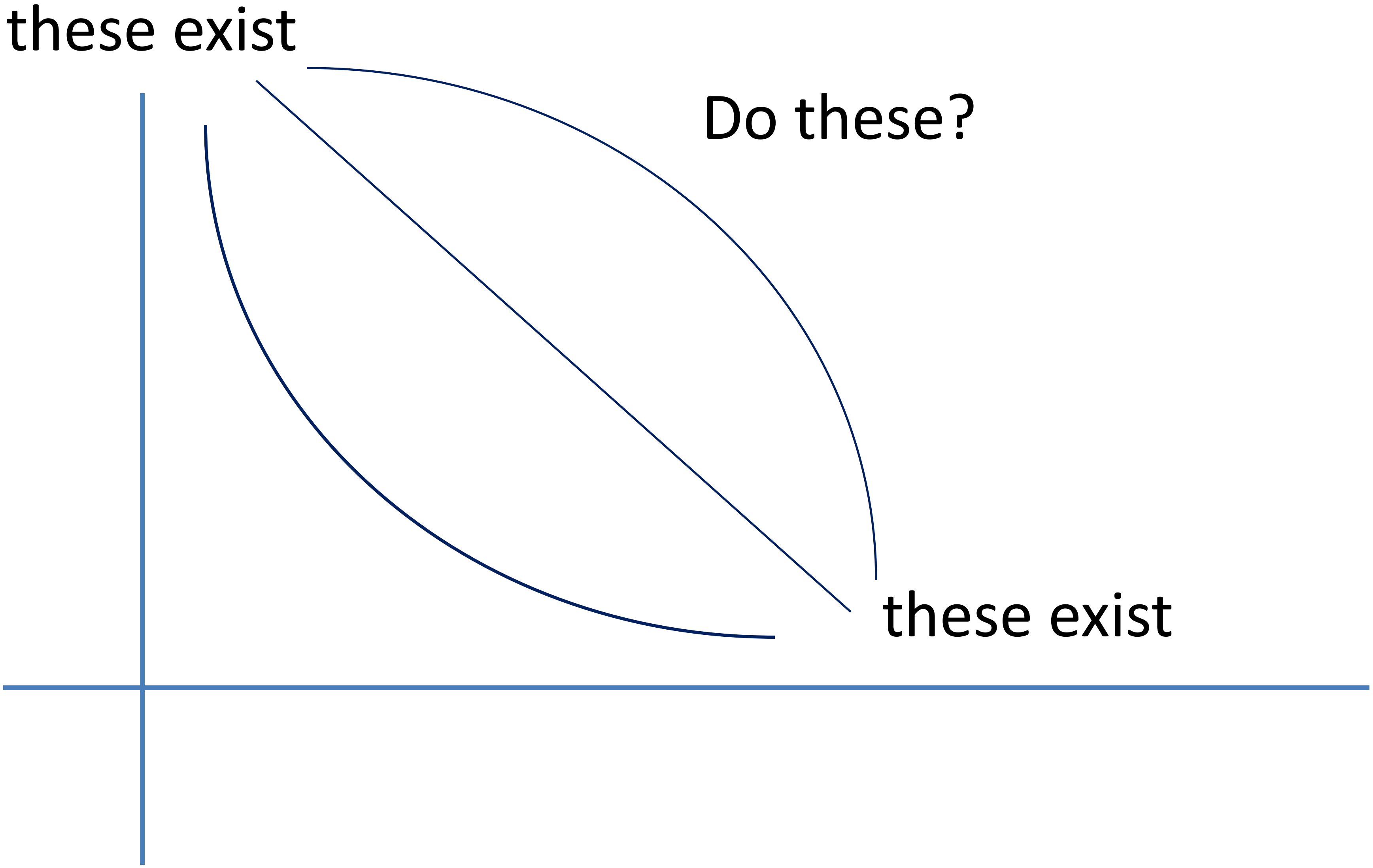
What is an HPC language?

- It’s a language that routinely achieves high performance on high performance hardware

- Conventional view: It don’t mean a thing if it ain’t got that exaflop.

- Should we begin to modify this view?

Performance and Programmability: Which curve are we on?

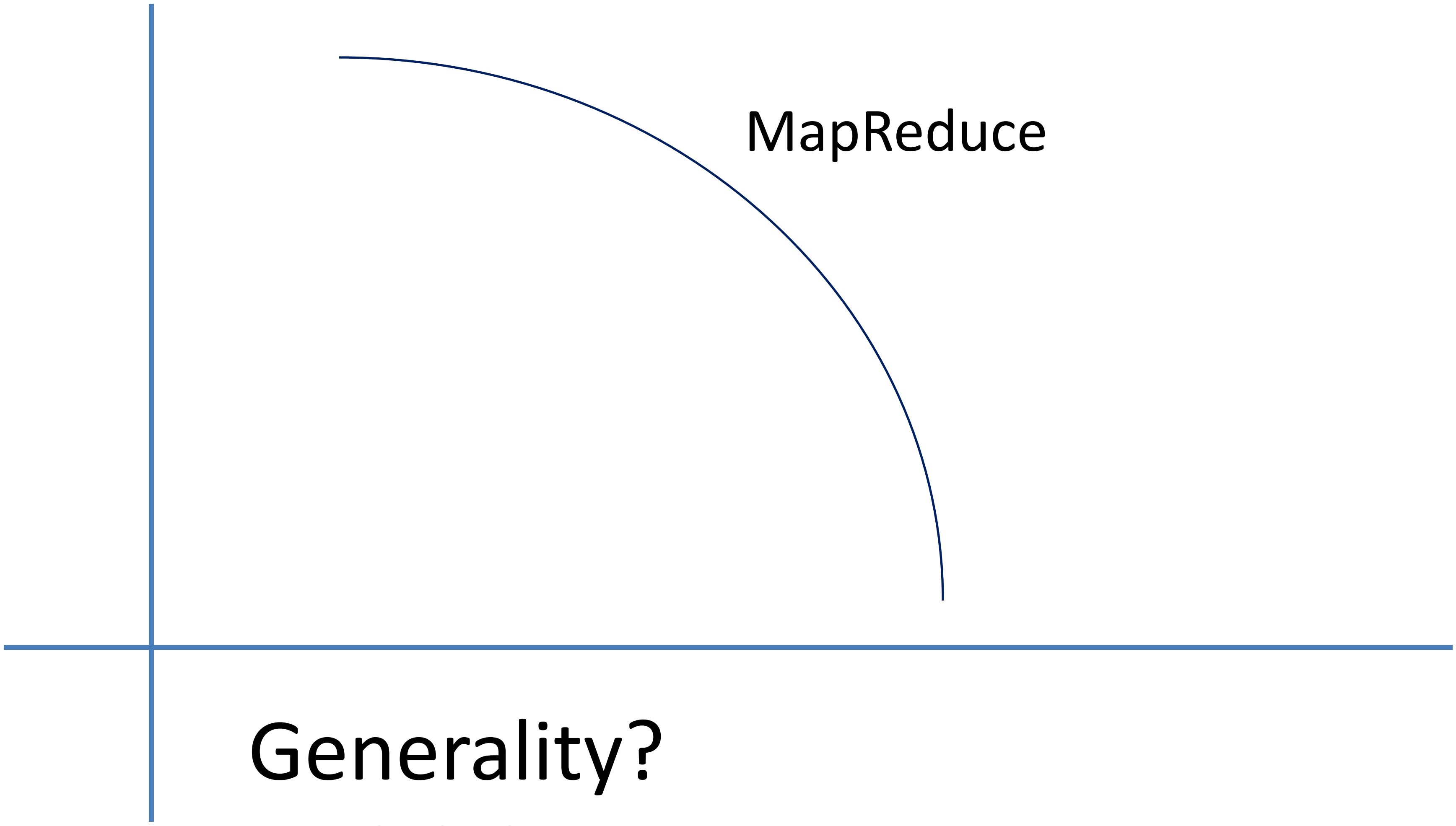


these exist

Do these?

these exist

Scale and Programmability



Generality?

Scalability?

Many ilities:
Which of them are simultaneously achievable?

- Performance, Programmability, Scale
- Maybe not, in general

Some Attempts

- HPF
- Linda
- PGAS
- Charm
- Parallel Matlab
- the slide is not big enough to contain the whole list

HPF – An early attempt

- High Performance Fortran
- Simple model
- Complex metalanguage of mappings
- Single thread of control
- A huge implementation job

Linda

- SPMD with tuple space
- Two levels of storage, local and shared
- The shared layer is a machine independent abstraction
- Performance
- Scalability

PGAS

- SPMD with distributed arrays, global indexes, global pointers
- More natural than put/get, shmem, MPI one sided
- A fairly low level language → performance isn't bad!
- Shared memory has its own issues
- Why isn't it more popular?

Charm

- Overdecomposition, virtualizing the node
- The actor approach
- Performance not bad
- Same question as for PGAS

Parallel Matlab

- Not a simple parallelization of the Matlab array operations
- A nice way to encapsulate the MPI collective communications
- Goal is high performance and scalability compared to Matlab on one core -- not scalable to biggest problems

Closing thoughts

- The problem is getting harder
- No one approach is best overall
- Big data and cloud create an even bigger market for HPC clusters
- Just as in programming generally, we will have to accept many approaches, and will build multi-language software