High-Performance Design Patterns for Modern Fortran

Variation points

- Accomodate for change
- Partial differential equation solvers
- Which coordinate system? How many dimensions?

Coordinate-free programming

$$\frac{\partial \vec{u}}{\partial t} = \nu \nabla^2 \vec{u} - \vec{u} \cdot \nabla \vec{u}.$$

```
class(tensor):: u_t, u
real:: nu = 1.0
u_t = nu * (.laplacian.u) - (u.dot.(.grad.u))
```

- Independent of dimension, coordinate systems etc.
- Mathematically and computationally precise.

Modern fortran

- Class abstraction
- Array operations
- Coarrays
- Prohibits function results containing coarrays

```
X = [sin(A + B) * C, 0., 1., 2., 3., 4., 5.];
X = X(1:5).
```

```
real, allocatable:: a(:,:,:)[:]
```

```
if (this_image() == 3) then
  a(1, 1, 1)[1] = a(1, 1, 1)[2]
end if
```

Design patterns

- Object superclass and error tracing

- Compute globally, return locally

```
type, abstract:: object
   logical:: user_defined = .false.
contains
   procedure:: is_defined
   procedure:: mark_as_defined
end type
```

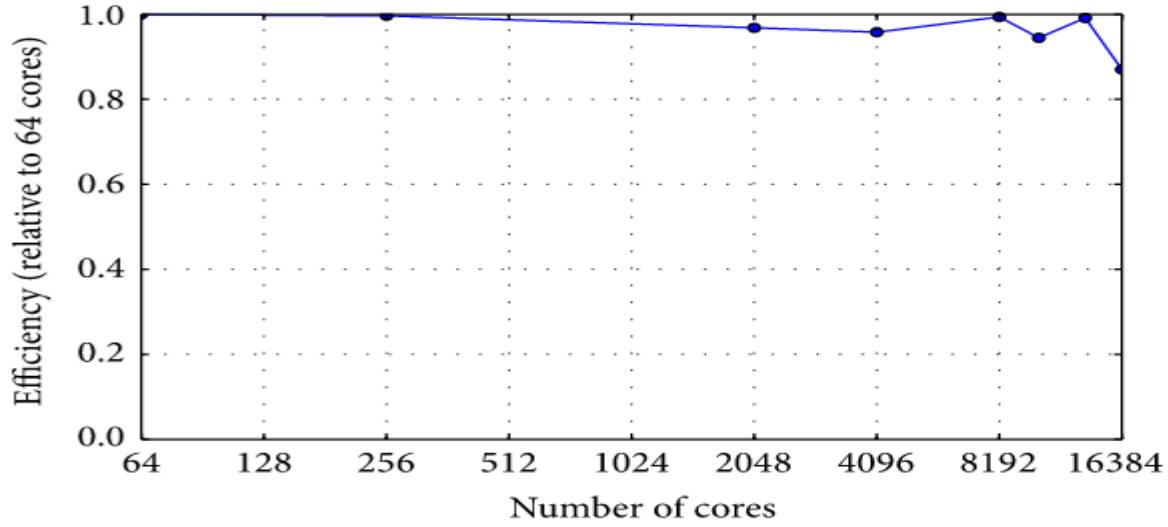
Pattern tradeoffs

- Object pattern
  - Lightweight performance
  - Heavyweight in source code writings

- Compute globally return locally

Weak scaling

- One dimensional burgers equation $u_t = \nu u_{xx} - u u_x.$
- Weak scaling coarray on Cray:
- Each core is assigned a fixed data size of 2 097 152 values for 3 000 time steps, syncing each time step

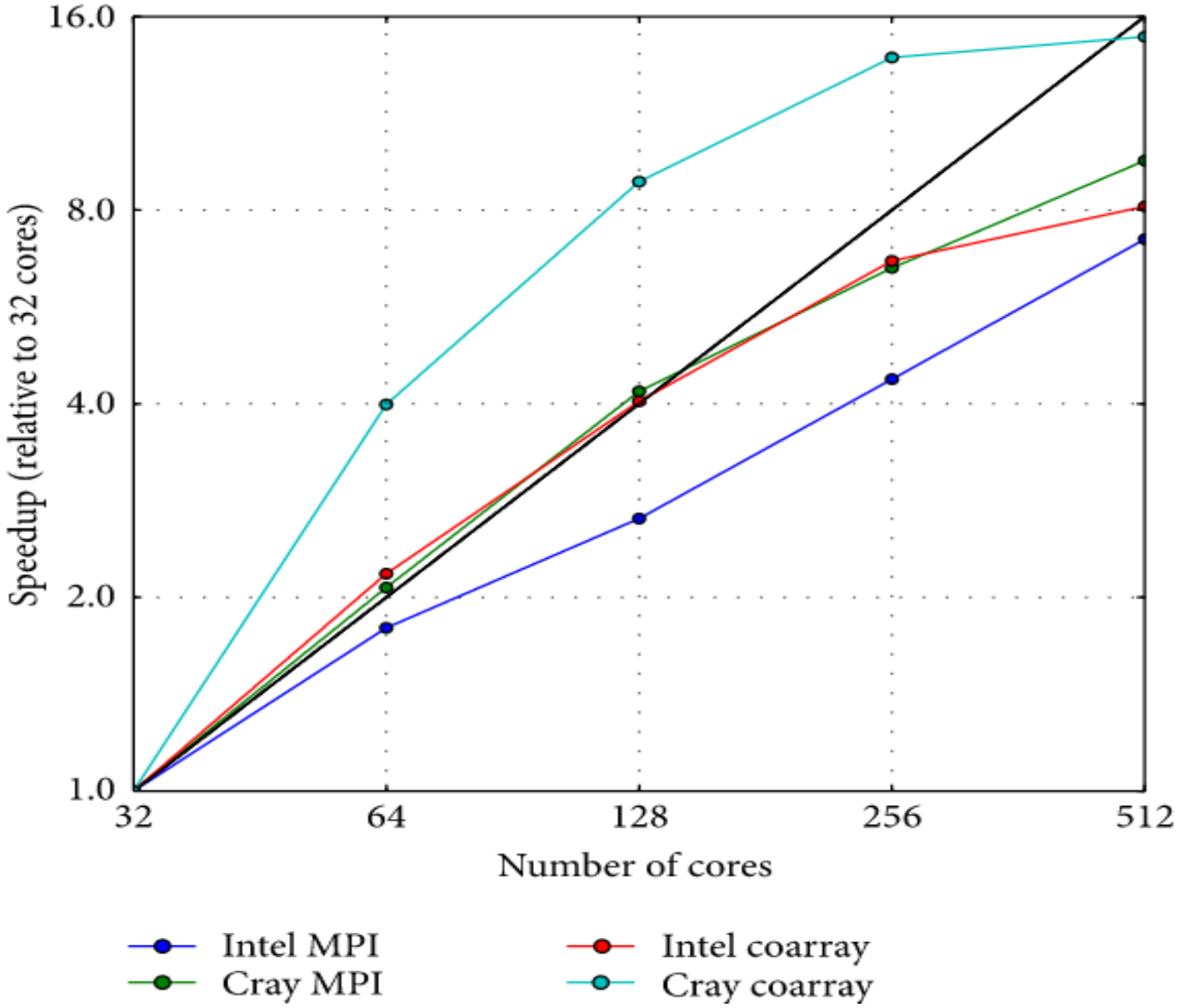


Cores	Time (s)	Efficiency
64	20414	100.0
256	20486	99.6
2048	21084	96.8
4096	21311	95.8
8192	20543	99.4
10240	21612	94.5
13312	20591	99.1
16384	23461	87.0

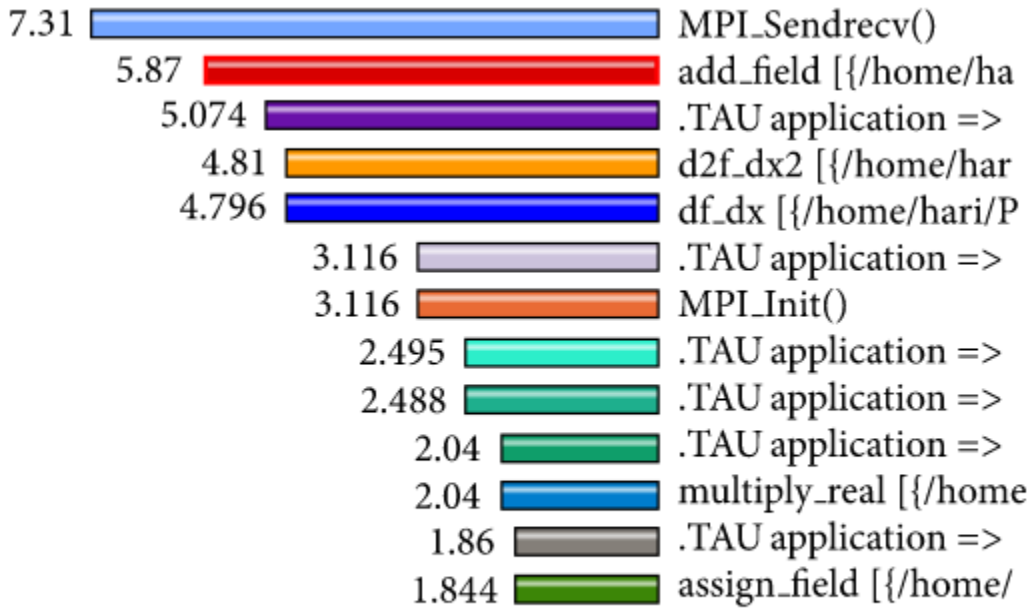
Strong scaling

- 409600 grid points
- MPI faster
- Superlinear speedup

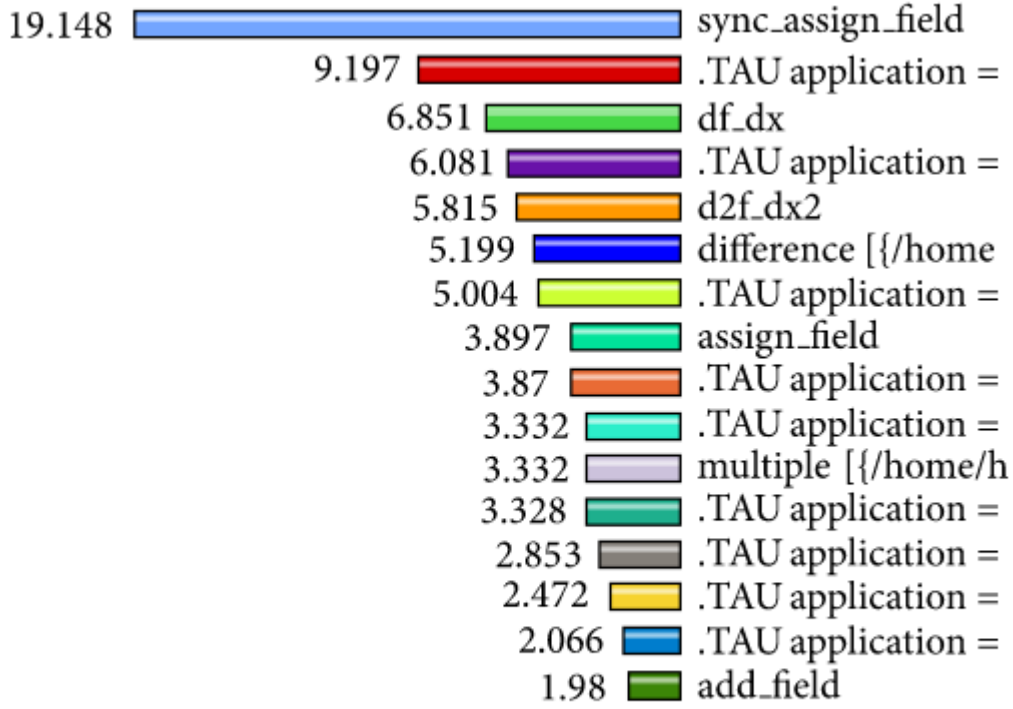
	409600 grid points			
Cores	MPI		CAF	
	Intel	Cray	Intel	Cray
32	52.675	59.244	154.649	187.204
64	29.376	28.598	71.051	46.923
128	19.864	14.169	38.321	21.137
256	12.060	9.109	23.183	13.553
512	7.308	6.204	19.080	12.581



Execution profiles



(a) MPI execution profile



(b) CAF execution profile

- Communication most expensive.
- Sync more expensive than Sendrecv.

Code complexity

- MPI more complex

Table 2: Code complexity of CAF versus MPI.

Metric	CAF	MPI
LOC	238	326
Use statements	3	13
Variables declared	58	97
External calls	0	24
Function arguments	11	79