This is not Your Parents' Fortran: A Scalable, Parallel, Functional OO PDE Solver

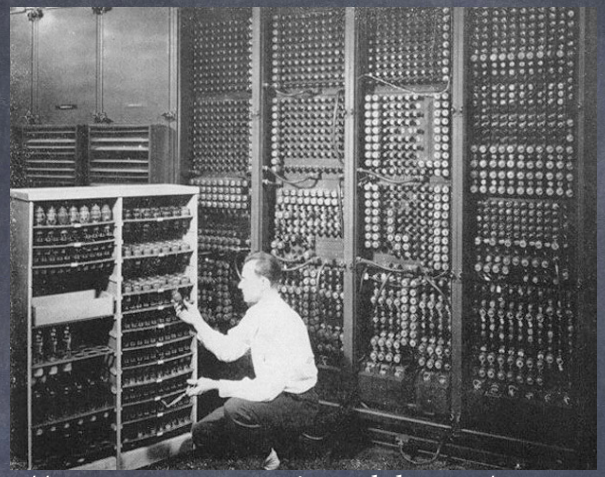
Damian Rouson

STANFORD UNIVERSITY
CENTER FOR COMPUTATIONAL EARTH AND ENVIRONMENTAL SCIENCES

The Bottom Line

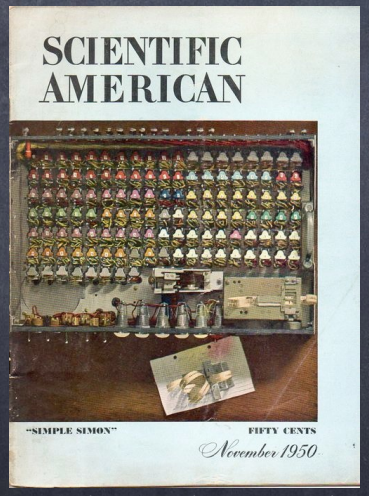
- In 1 academic quarter, I teach beginning graduate students how to write a
  - Parallel
  - Functional
  - Object-Oriented
- PDE solver using high-level mathematical abstractions,
- Scaling beyond 16,000 cores with nearly 90% parallel efficiency,
- Using either of 2 commercially released compilers or one pre-release open-source compiler,
- With no reliance on libraries external to the language (e.g., no OpenMP or MPI in the source)
- With zero chance of common beginner mistakes (e.g., no memory leaks or dangling pointers).

Fortran's Image



http://www.computersciencelab.com/ComputerHistory/HistoryPt4.htm

http://www.computersciencelab.com/ComputerHistory/HistoryPt4.htm



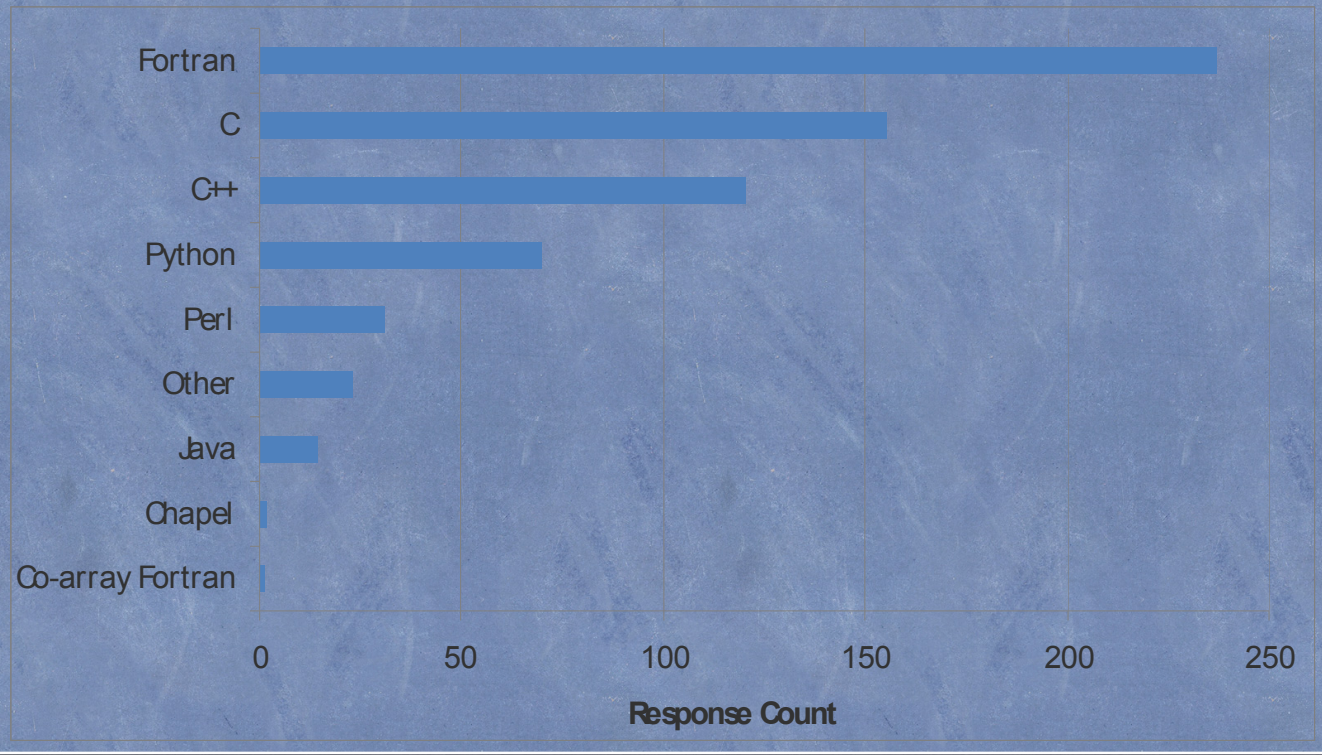
http://longstreet.typepad.com/thesciencebookstore/computer_techhistory/

http://www.clemson.edu/caah/history/facultypages/PamMack/lec122sts/computers.html

Fortran's Reality

Question 31: Which programming models and languages do you use for code development? Please select one or more from the following list.

Response rate: 78%



Source: Bull, M., Guo, X., Ioannis Liabotis, I. (Feb. 2011) *Applications and user requirements for Tier-0 systems*, PRACE Consortium.

Fortran's Future

8. Please indicate your requirements for comprehensive formal training in the following programming languages:

	Not important	Somewhat important	Very important	Rating Average	Response Count
FORTRAN77	**75.9% (60)**	16.5% (13)	7.6% (6)	0.32	79
Fortran 95	31.8% (27)	**41.2% (35)**	27.1% (23)	0.95	85
Fortran 2003	34.9% (29)	**36.1% (30)**	28.9% (24)	0.94	83
C	**37.5% (33)**	35.2% (31)	27.3% (24)	0.90	88
C++	**38.9% (35)**	33.3% (30)	27.8% (25)	0.89	90
Java	**78.1% (57)**	17.8% (13)	4.1% (3)	0.26	73
Scripting Languages (Python, PERL, Ruby etc.)	39.1% (34)	**41.4% (36)**	19.5% (17)	0.80	87
If you use another language, please indicate the relative importance of training in it:					7
				answered question	**97**
				skipped question	**22**

Source: Stitt, T. and T. Robinson (2008) *A Survey on Training and Education Needs for Petascale Computing*, PRACE Consortium Partners (http://www.tinyurl.com/PRACE-survey-2008).

“We don’t believe that … Joe the programmer should have to deal with parallelism in an explicit way”

Kunle Olukotun

Going parallel should not mean writing low-level parallel code, just as going high-performance need not require assembly language (as in the good old days) compilers should help, even though, this is a difficult task.
A programming model that presents parallelism to the programmer in a simple yet powerful way can achieve surprisingly good results with the proper compiler support. While at the same time comfortably outperform what most novice programmers would produce with great effort when forced to exercise direct control.

Compiler Support

Compiler	OOP+Functional	Parallel
Cray	x	x
Intel	x	x
GNU	x	o
IBM	x	
Portland Group	x	
NAG	x	

- "x" -> a released version supports all features employed in this talk.
- "o" -> a pre-release version supports all features employed in this talk.

Parallel Functional OOP in Modern Fortran

OOP —> Functional —> Parallel

Parallel programming (Fortran 2008)

Object-oriented programming (Fortran 2003)

Functional Programming (Fortran 95)

User-defined, purely functional operators

```
u_t = -(.grad.p)/rho + nu*(.laplacian.u) -(u.dot.(.grad.u))
```

Distributed objects containing coarrays

"Do Concurrent"

```
program main
  implicit none
  integer, parameter :: num_particles=10,num_dimensions=3
  integer :: stride
  real :: V(num_particles,num_dimensions),response_time(num_particles)
  V=100.
  stride = input_from_file()
  do concurrent(particle=1:num_particles:stride)
    V(particle,:) = &
      V(particle,:) - dt*V(particle,:)/response_time(particle)
    ! Can also call pure procedures here
  end do
end program
```

- Supporting compiler technologies:
  - Gfortran: SIMD via front-end pragmas
  - Intel: Vectorization (AVX)

Case Study: Morfeus

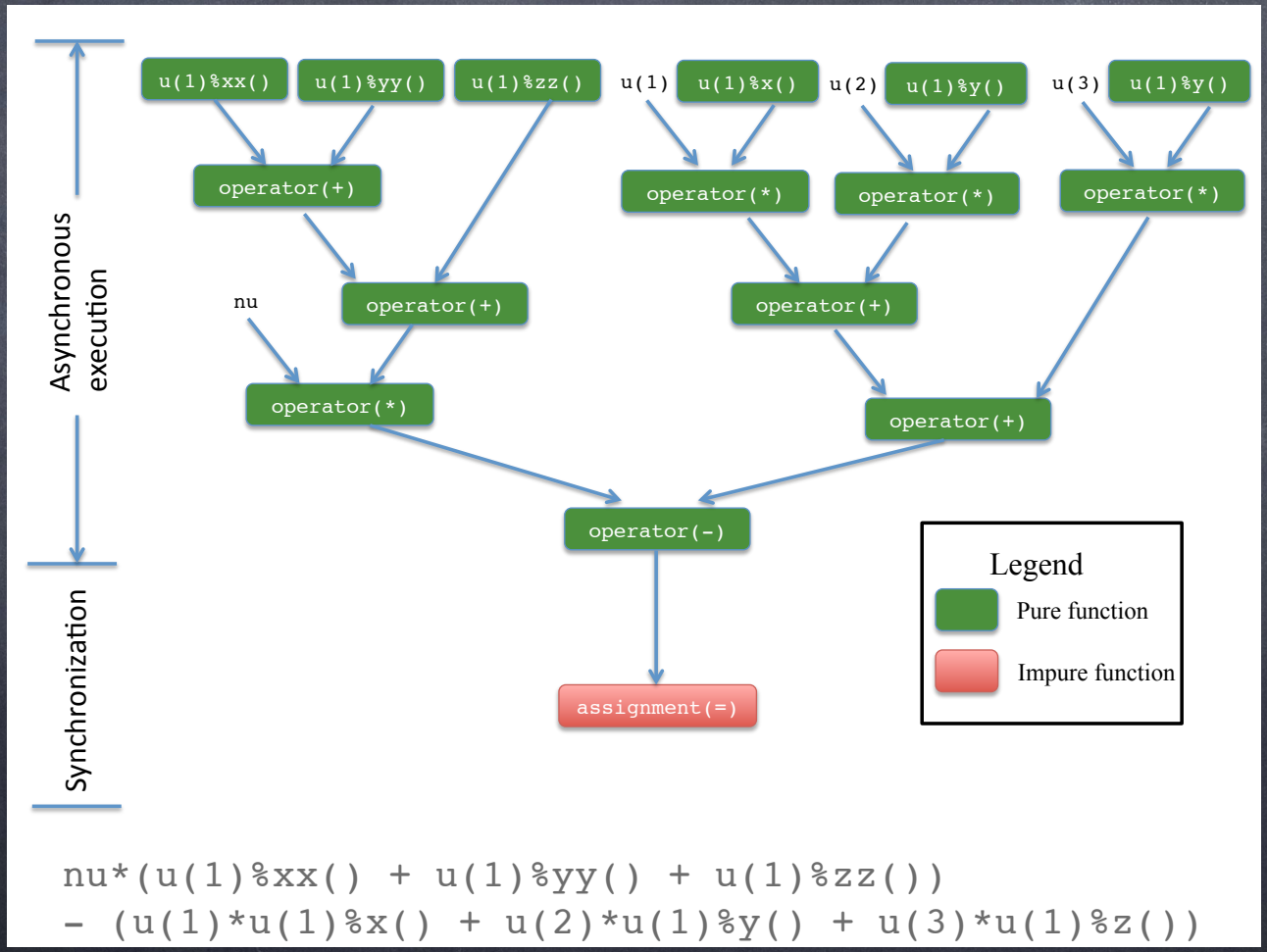
- A coordinate-free PDE solver framework:

```
program main
  use cartesian_tensor_class, only : cartesian_tensor
  use scalar_field_class, only : scalar_field
  implicit none
  type(cartesian_tensor) :: u
  real :: t=0.,dt=0.1,t_final=1.0,nu=0.01
  type(scalar_field) :: initial(3)
  u=cartesian_tensor(initial,rank=1,space_dim=3,covariant=[.true.])
  do while(t<t_final)
```

$$\vec{u}^{n+1} = \vec{u}^{n} + \Delta t \left(\nu \nabla^2 \vec{u}^{n} - \vec{u}^{n} \cdot \nabla \vec{u}^{n} \right)$$

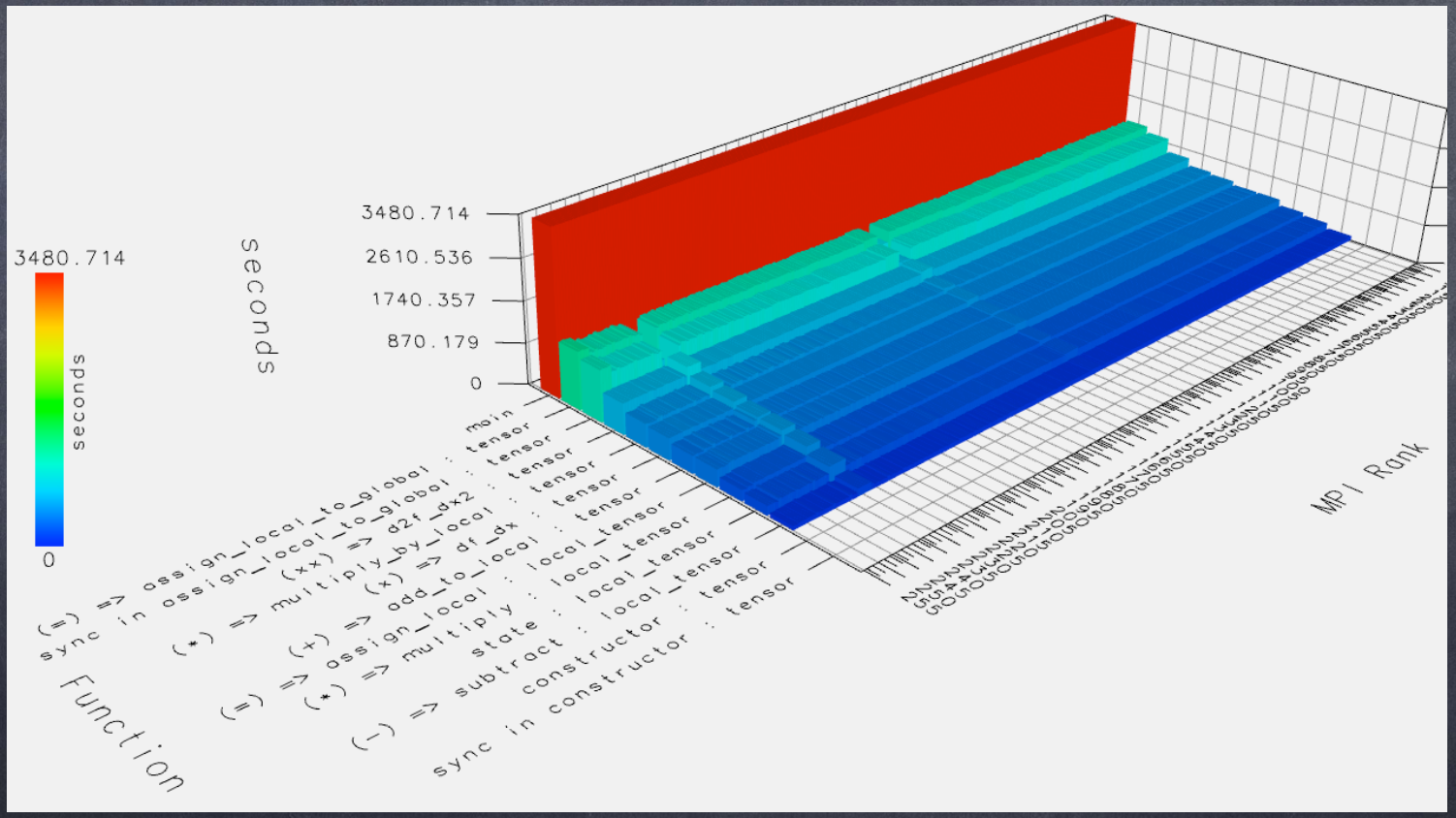
```
    u = u + dt*( nu*(.laplacian.u) - (u.dot.(.grad.u)) )
    t = t + dt
  end do
end program
```

Asynchronous Expression Evaluation

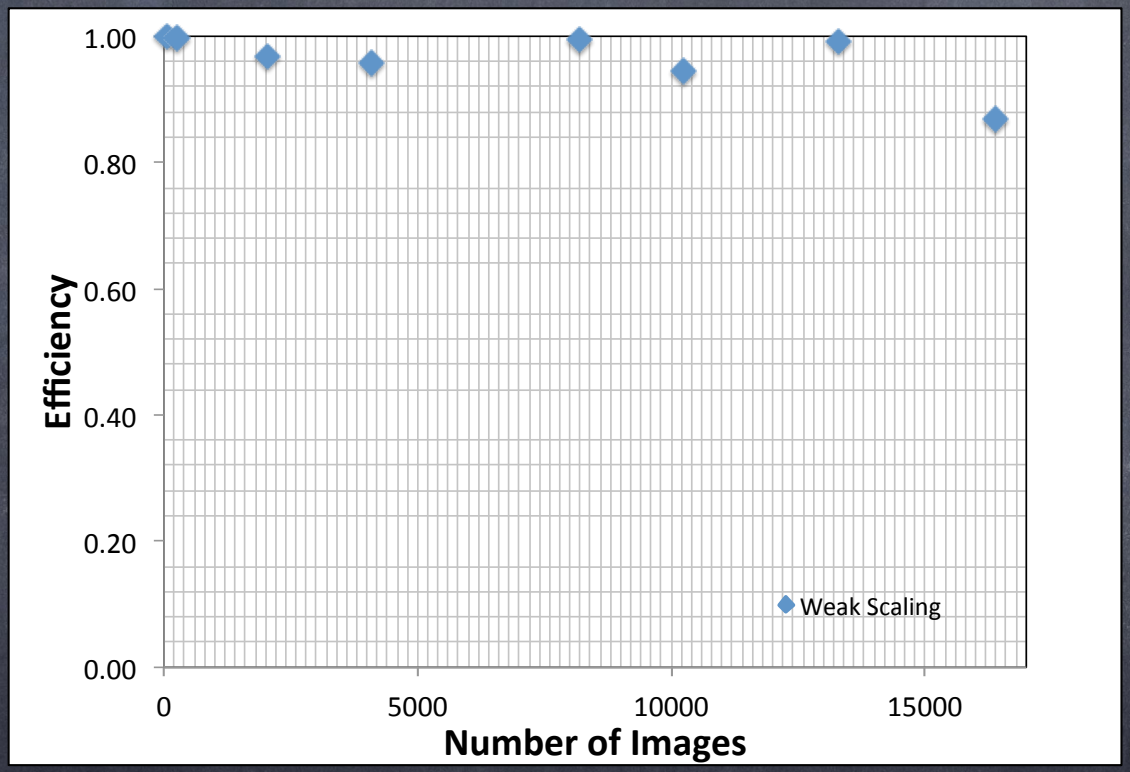



```
nu*(u(1)%xx() + u(1)%yy() + u(1)%zz())
- (u(1)*u(1)%x() + u(2)*u(1)%y() + u(3)*u(1)%z())
```

1D Burgers Solver Load Balance



Results: 1D Burgers Equation Weak Scaling



Fortran Philosophy

"Communicate properties, not optimizations."

Conclusions

- Fortran is now a PGAS language with a platform-agnostic, scalable parallel programming model.
- Modern Fortran supports multiple programming paradigms that fully integrate with its PGAS features: array programming, functional programming, object-oriented programming.
- Very broad support for OOP/Functional programming features exists.
- Parallel programming feature support is growing.
- Productivity for beginners is high.

References

- Haveraaen, M., K. Morris, and D. W. I. Rouson (2013) "High-performance design patterns for modern Fortran," First International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering, Denver, Colorado, USA. November 22.

- Radhakrishnan, H., D. W. I. Rouson, K. Morris, S. Shende, and S. C. Kassinos (2013) "Test-driven coarray parallelization of a legacy Fortran application," First International Workshop on Software Engineering High Performance Computing in Computational Science and Engineering, Denver, Colorado, USA. November 22.

Acknowledgements

- Karla Morris, Sandia National Laboratories
- Hari Radhakrishnan and Stavros Kassinos, University of Cyprus
- Magne Haveraaen, University of Bergen
- Jim Xia, IBM
- Xiaofeng Xu, GM
- Sameer Shende, University of Oregon