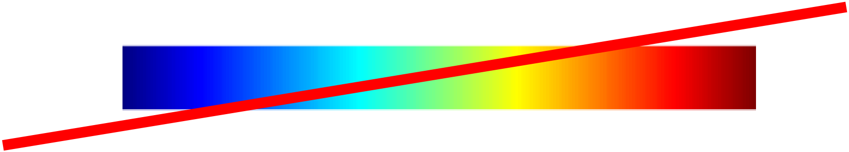
http://www.climate-lab-book.ac.uk/
2014/end-of-the-rainbow/



Firedrake: automating the finite element method by composing abstractions

Lawrence Mitchell[1]

6th July 2016

[1]Departments of Computing and Mathematics, Imperial College London

Firedrake team

IC	David A. Ham, Miklós Homolya, Fabio Luporini, Gheorghe-Teodor Bercea, Paul H. J. Kelly
Bath	Andrew T. T. McRae
ECMWF	Florian Rathgeber

`www.firedrakeproject.org`

Rathgeber et al. 2015 `arXiv:1501.01809 [cs.MS]`

The right abstraction level

A specification of finite element problems

```python
from firedrake import *
mesh = UnitSquareMesh(100, 100)
V = FunctionSpace(mesh, "RT", 2)
Q = FunctionSpace(mesh, "DG", 1)
W = V*Q
u, p = TrialFunctions(W)
v, q = TestFunctions(W)

a = dot(u, v)*dx + div(v)*p*dx + div(u)*q*dx
L = -Constant(1)*v*dx
u = Function(W)
solve(a == L, u, solver_parameters={
    "ksp_type": "gmres",
    "ksp_rtol": 1e-8,
    "pc_type": "fieldsplit",
    "pc_fieldsplit_type": "schur",
    "pc_fieldsplit_schur_fact_type": "full",
    "pc_fieldsplit_schur_precondition": "selfp",
    "fieldsplit_0_ksp_type": "preonly",
    "fieldsplit_0_pc_type": "ilu",
    "fieldsplit_1_ksp_type": "preonly",
    "fieldsplit_1_pc_type": "hypre"
})
```

Find $u \in V \times Q \subset H(\mathrm{div}) \times L^2$ s.t.

$$\langle u, v\rangle + \langle \mathrm{div}\, v, p\rangle = 0 \quad \forall v \in V$$

$$\langle \mathrm{div}\, u, q\rangle = -\langle 1, q\rangle \quad \forall q \in Q.$$

More than a pretty face

Library usability

- High-level language enables rapid model development
- Ease of experimentation
- Small model code base

Library development

- Automation of complex optimisations
- Exploit expertise across disciplines
- Small library code base

www.dolfin-adjoint.org

Automated derivation of the discrete adjoint from forward models written using FEniCS.

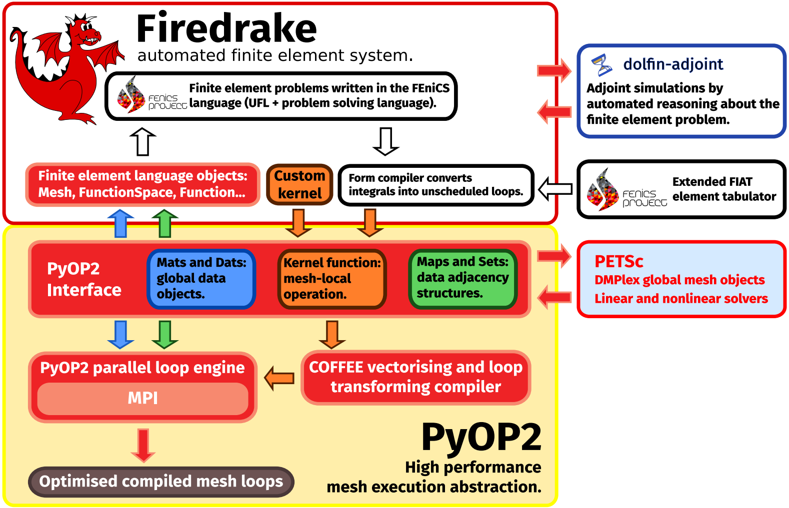
```
$ cloc dolfin-adjoint/
Language   files    blank    comment    code
Python        52     2228        878    6939
$ cloc dolfin-adjoint/compatibility.py
Python         1       36          9     135
```

How much code do you need to change to

- Change preconditioner (e.g. ILU to AMG)?
- Drop terms in the preconditioning operator?
- Use a completely different operator to precondition?
- Do quasi-Newton with an approximate Jacobian?
- Apply operators matrix-free?

Same "easy to use" code must run fast at scale.

Say *what*, not *how*.


Firedrake
automated finite element system.
Finite element problems written in the FEniCS language (UFL + problem solving language).
Finite element language objects: Mesh, FunctionSpace, Function...
Custom kernel
Form compiler converts integrals into unscheduled loops.
dolfin-adjoint
Adjoint simulations by automated reasoning about the finite element problem.
Extended FIAT element tabulator
PyOP2 Interface
Mats and Dats: global data objects.
Kernel function: mesh-local operation.
Maps and Sets: data adjacency structures.
PETSc
DMPlex global mesh objects
Linear and nonlinear solvers
PyOP2 parallel loop engine
MPI
COFFEE vectorising and loop transforming compiler
PyOP2
High performance mesh execution abstraction.
Optimised compiled mesh loops

Local kernels

Problem

Modern optimising compilers do a bad job on finite element kernels.

Problem

Modern optimising compilers do a bad job on finite element kernels.

Code motion (or not?)

```
for (i = 0; i < L; i++ )
   for (j = 0; j < M; j++)
      for (k = 0; k < N; k++)
         A[j][k] += f(i, j)*g(i, k)
```

Problem

Modern optimising compilers do a bad job on finite element kernels.

Code motion (or not?)

```
for (i = 0; i < L; i++ )
   for (j = 0; j < M; j++)
      for (k = 0; k < N; k++)
         A[j][k] += f(i, j)*g(i, k)
```

Corollary

We need to spoon-feed the compiler already optimised code.

A job for an expert

Hardware-aware optimsation of finite element kernels is a job for:

A job for an expert

Hardware-aware optimsation of finite element kernels is a job for:

- A numerical analyst?

A job for an expert

Hardware-aware optimsation of finite element kernels is a job for:

- A numerical analyst?
- A geodynamicist?

A job for an expert

Hardware-aware optimsation of finite element kernels is a job for:

- A numerical analyst?
- A geodynamicist?
- A computational chemist?

A job for an expert

Hardware-aware optimsation of finite element kernels is a job for:

- A numerical analyst?
- A geodynamicist?
- A computational chemist?
- A computational scientist?

A job for an expert

Hardware-aware optimsation of finite element kernels is a job for:

- A numerical analyst?
- A geodynamicist?
- A computational chemist?
- A computational scientist?
- A computer scientist?

Automating expertise

- "In-person" case-by-case optimisation *does not scale*
- Code generation allows us to package expertise and provide it to everyone
- Done by a special-purpose kernel compiler

COFFEE I

No single optimal schedule for evaluation of every finite element kernel. Variability in

- polynomial degree,
- number of fields,
- kernel complexity,
- working set size,
- structure in the basis functions,
- structure in the quadrature points,
- ...

Imperial College London

Vectorisation

Align and pad data structures, then use intrinsics or rely on compiler.

Luporini, Varbanescu, et al. 2015 `doi:10.1145/2687415`

Flop reduction

Exploit *linearity* in test functions to perform factorisation, code motion and CSE.

Luporini, Ham, and Kelly 2016 `arXiv:1604.05872 [cs.MS]`

`github.com/coneoproject/COFFEE`

Global iteration

Performance

- Keep data in cache as long as possible.
- Manually fuse kernels.
- Loop tiling for latency hiding.
- ...
- Individual components hard to test
- Space of optimisations suffers from combinatorial explosion.

Maintainability

- Keep kernels separate
- “Straight-line” code
- ...
- Testable
- Even if performance of individual kernels is good, can lose *a lot*

PyOP2

A library for expressing data parallel iterations

Sets	iterable entities
Dats	abstract managed arrays (data defined on a set)
Maps	relationships between elements of sets
Kernels	local computation
par_loop	Data parallel iteration over a set

Arguments to parallel loop indicate how to gather/scatter global data using *access descriptors*

```
par_loop(kernel, iterset, data1(map1, READ), data2(map2, WRITE))
```

Local computation

Kernels do not know about global data layout.

- Kernel defines contract on local, packed, ordering.
- Global-to-local reordering/packing appears in map.

"Implicit" iteration

Application code does not specify explicit iteration order.

- Define data structures, then just "iterate"
- Lazy evaluation

Lazy evaluation

- `par_loop` only executed “when you look at the data”.
- PyOP2 sees sequence of loops, can reason about them for
    - Loop fusion
    - Loop tiling
    - Communication coalescing
- Application code does not change. “What, not how”.

Did we succeed?

Experimentation

With model set up, experimentation is easy

- Change preconditioner: c. 1 line
- Drop terms: c. 1-4 lines
- Different operator: c. 1-10 lines
- quasi-Newton: c. 1-10 lines
- Matrix-free: XXX

Core Firedrake

Component	LOC
Firedrake	9000
PyOP2	5000
TSFC	2700
COFFEE	4500
Total	21200

Shared with FEniCS

Component	LOC
FIAT	4000
UFL	13000
Total	17000

Performance I

Kernel performance

- COFFEE produces kernels that are better (operation count) than existing automated form compilers
- Provably optimal in some cases
- Good vectorised performance, problem dependent, but up to 70% peak for in-cache computation.

Thetis

- 3D unstructured coastal ocean model written with Firedrake
- 5000 LOC, c. 1 person year
- Lock exchange test case

Thetis	P1DG-P1DG, triangular wedges. 24 s/s.
SLIM	hand-coded/optimised (same numerics), 6 s/s

`github.com/thetisproject/thetis`

Summary

- Firedrake provides a layered set of abstractions for finite element
- Enables automated provision of expertise to model developers
- Computational performance is good, often $> 50\%$ achievable peak.
- Hero-coding necessary if you want the last 10-20%
- ...but at what (person) cost?

Imperial College London

Want to work on FEM at Imperial? We are hiring.

Questions?

References

Luporini, F., D. A. Ham, and P. H. J. Kelly (2016). *An algorithm for the optimization of finite element integration loops*. Submitted. arXiv: 1604.05872.

Luporini, F., A. L. Varbanescu, et al. (2015). "Cross-Loop Optimization of Arithmetic Intensity for Finite Element Local Assembly". *ACM Trans. Archit. Code Optim.* **11**. doi:10.1145/2687415.

Rathgeber, F. et al. (2015). *Firedrake: automating the finite element method by composing abstractions*. Submitted. arXiv: 1501.01908.

Imperial College London