Code Specialisation of Auto Generated GPU Kernels

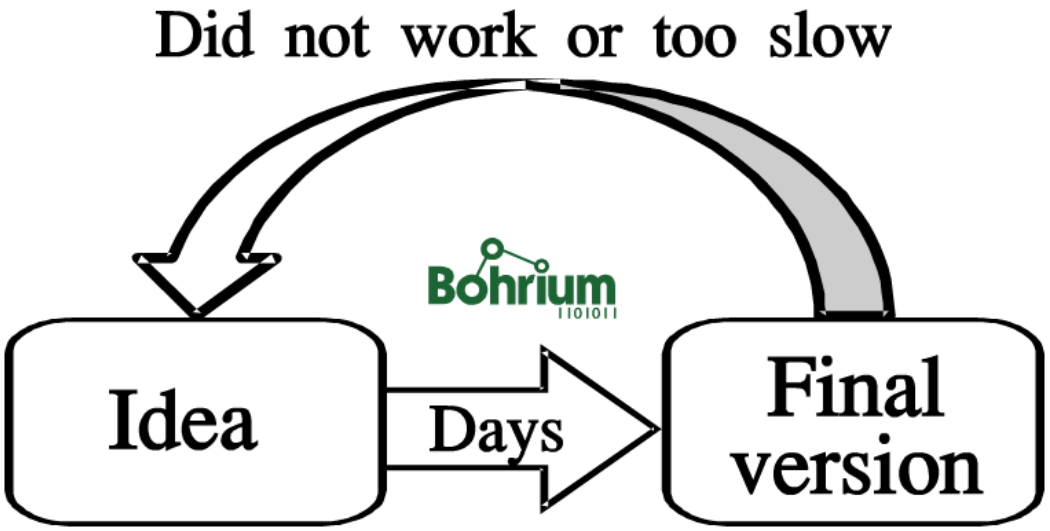
Troels Blum

University of Copenhagen
DENMARK

http://bh107.org

Development Cycle

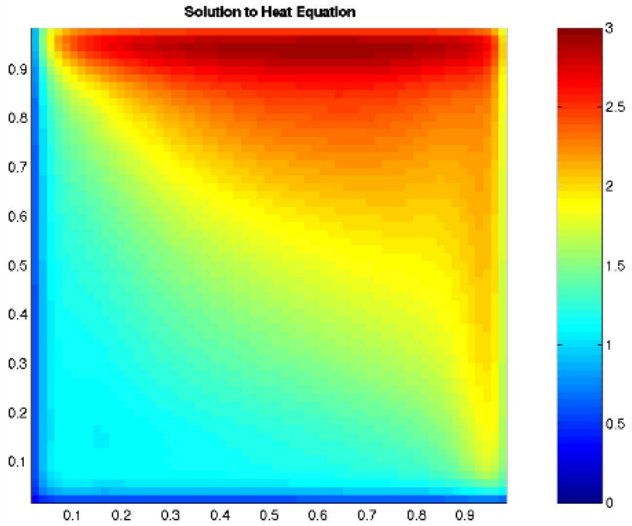
High-Performance vs High-Productivity



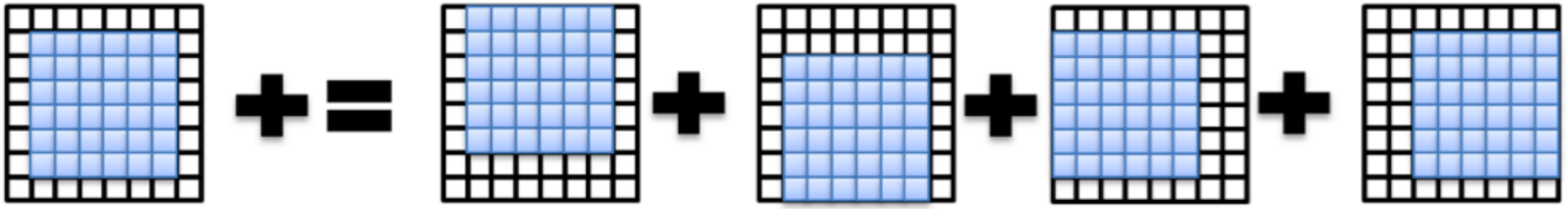
Array Programming

2D 5-point stencil

```
def heat_eq(grid, epsilon=0.005):
    delta  = epsilon + 1
    while delta > epsilon:
        work = (grid[1:-1,1:-1] + grid[0:-2,1:-1] +
                grid[1:-1,2:] + grid[1:-1,0:-2] +
                grid[2:,1:-1]) * 0.2
        delta = numpy.sum(numpy.absolute(work - grid[1:-1,1:-1]))
        grid[1:-1,1:-1] = work
    return grid
```



(by Dr. Kropinski)

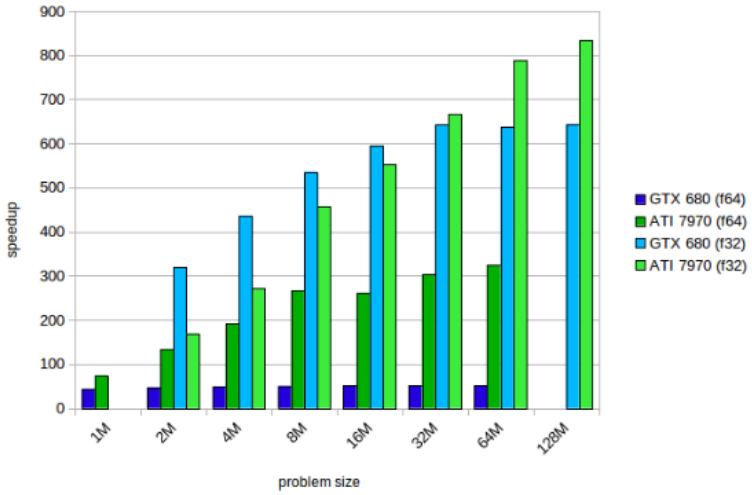


A[1:-1,1:-1] += A[:-2, 1:-1] + A[2:, 1:-1] + A[1:-1,0:-2] + A[2:,1:-1]

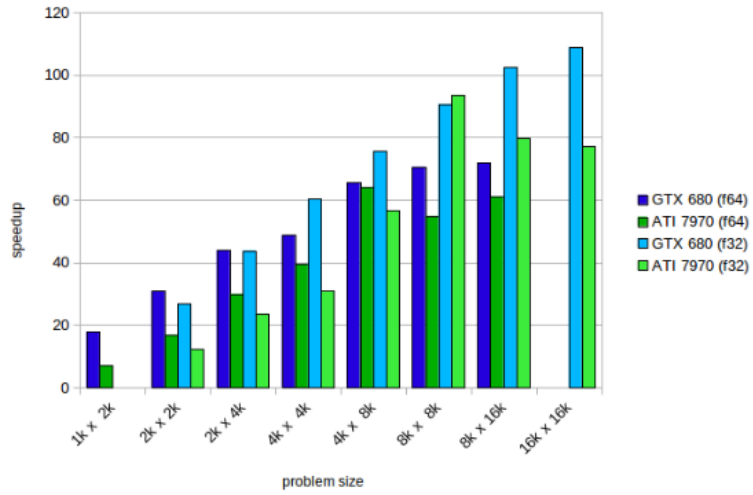
Benchmarks

Relative speedup compaired to native NumPy

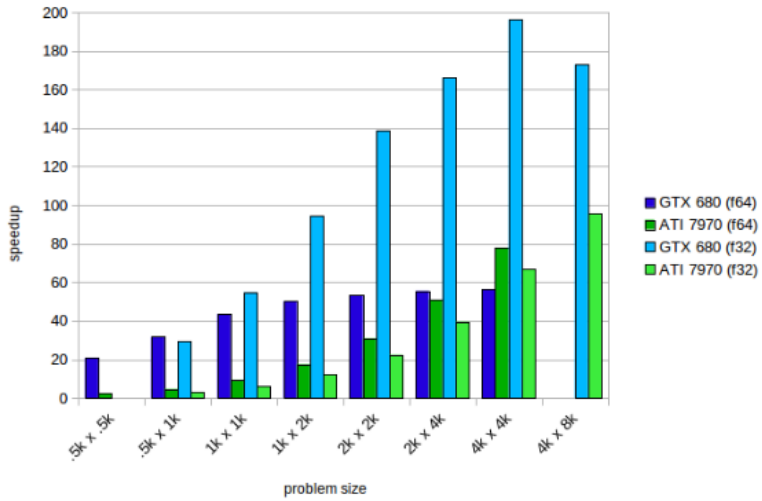
Black-Scholes



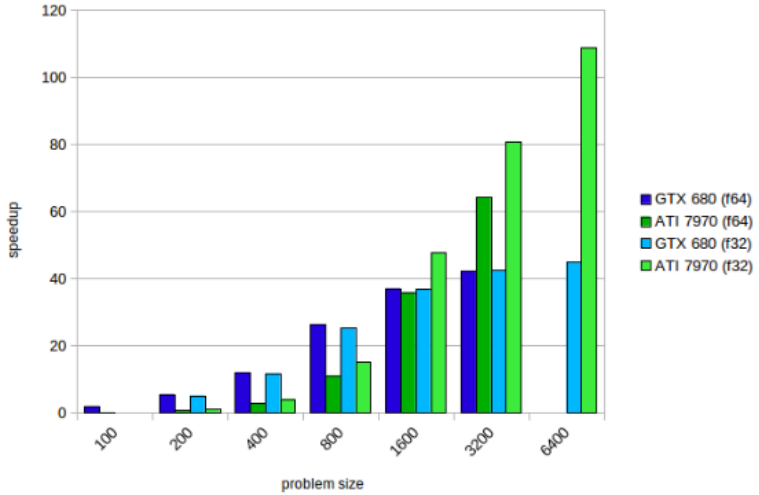
Successive Over-relaxation (SOR)



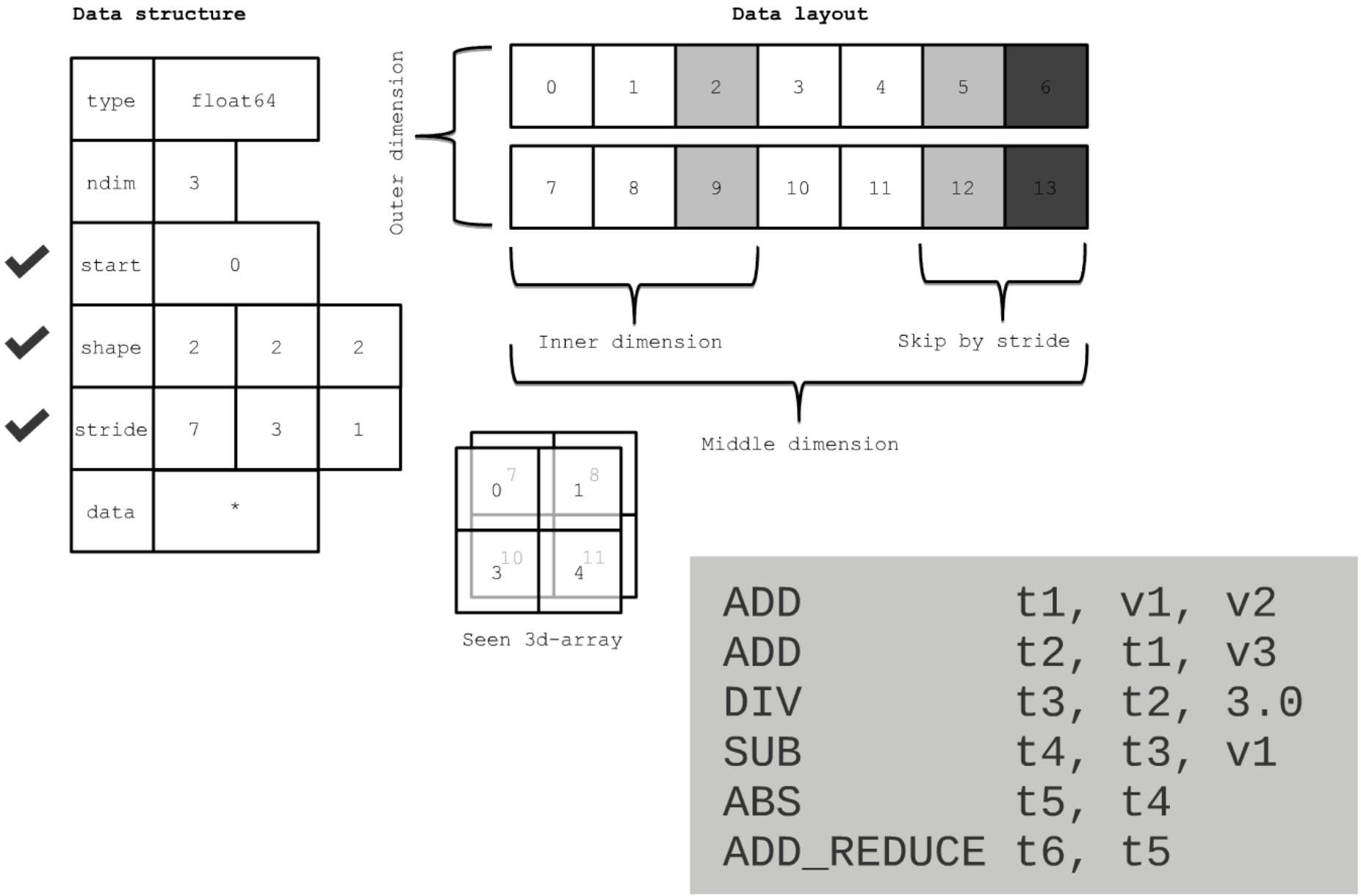
Shallow Water



N-body



Bohrium Byte-code




```
ADD          t1, v1, v2
ADD          t2, t1, v3
DIV          t3, t2, 3.0
SUB          t4, t3, v1
ABS          t5, t4
ADD_REDUCE   t6, t5
```

Generated GPU kernels

```
__kernel heat_eq (...)
{
        const size_t gidx = get_global_id(0);
        if (gidx >= ds0)
                return;
        const size_t gidy = get_global_id(1);
        if (gidy >= ds1)
                return;
        double v1 = a1[gidy*v1s2 + gidx*v1s1 + v1s0];
        double v2 = a1[gidy*v2s2 + gidx*v2s1 + v2s0];
        double v4 = a1[gidy*v4s2 + gidx*v4s1 + v4s0];
        double v6 = a1[gidy*v6s2 + gidx*v6s1 + v6s0];
        double v8 = a1[gidy*v8s2 + gidx*v8s1 + v8s0];
        double v0;
        v0 = v1 + v2;
        double v3;
        v3 = v0 + v4;
        double v5;
        v5 = v3 + v6;
        double v7;
        v7 = v5 + v8;
        double v9;
        v9 = v7 * s0;
        double v10;
        v10 = v9 - v1;
        double v11;
        v11 = fabs(v10);
        a5[gidy*v9s2 + gidx*v9s1 + v9s0] = v9;
        a7[gidy*v11s2 + gidx*v11s1 + v11s0] = v11;
}
```

```
__kernel heat_eq(...)
{
        const size_t gidx = get_global_id(0);
        if (gidx >= 1998)
                return;
        const size_t gidy = get_global_id(1);
        if (gidy >= 1998)
                return;
        double v1 = a1[gidy*2000 + gidx*1 + 2001];
        double v2 = a1[gidy*2000 + gidx*1 + 1];
        double v4 = a1[gidy*2000 + gidx*1 + 2002];
        double v6 = a1[gidy*2000 + gidx*1 + 2000];
        double v8 = a1[gidy*2000 + gidx*1 + 0];
        double v0;
        v0 = v1 + v2;
        double v3;
        v3 = v0 + v4;
        double v5;
        v5 = v3 + v6;
        double v7;
        v7 = v5 + v8;
        double v9;
        v9 = v7 * s0;
        double v10;
        v10 = v9 - v1;
        double v11;
        v11 = fabs(v10);
        a5[gidy*1998 + gidx*1 + 0] = v9;
        a7[gidy*1998 + gidx*1 + 0] = v11;
}
```

Assumptions

- Specialization will enhance kernel run-time
- Compilation of kernels is time consuming

Benchmark applications

- Stencil applications
  - 1D, 3 points
  - 2D, 9 points
  - 3D, 27 points
  - 4D, 81 points
- Gausian elimination
- LU decomposition

Kernel types

Dynamic
Fixed
Selected

```
def heat_eq(grid, epsilon=0.005):
    delta  = epsilon + 1
    while delta > epsilon:
        work = (grid[1:-1,1:-1] + grid[0:-2,1:-1] +
                grid[1:-1,2:] + grid[1:-1,0:-2] +
                grid[2:,1:-1]) * 0.2
        delta = numpy.sum(numpy.absolute(work - grid[1:-1,1:-1]))
        grid[1:-1,1:-1] = work
    return grid
```

1D - 4D stencil

```
def gauss(a):
    for c in xrange(1,a.shape[0]):
        a[c:,c-1:] = a[c:,c-1:] -
                     (a[c:,c-1]/a[c-1,c-1:c])[:,None] *
                     a[c-1,c-1:]
    a /= numpy.diagonal(a)[:,None]
    return a
```

Gauss elimination

```
def lu(a):
    u = a.copy()
    l = numpy.zeros_like(a)
    numpy.diagonal(l)[:] = 1.0
    for c in xrange(1,u.shape[0]):
        l[c:,c-1]  = u[c:,c-1] / u[c-1,c-1:c]
        u[c:,c-1:] = u[c:,c-1:] -
                     l[c:,c-1][:,None] * u[c-1,c-1:]
    return (l,u)
```

1D - 4D stencil

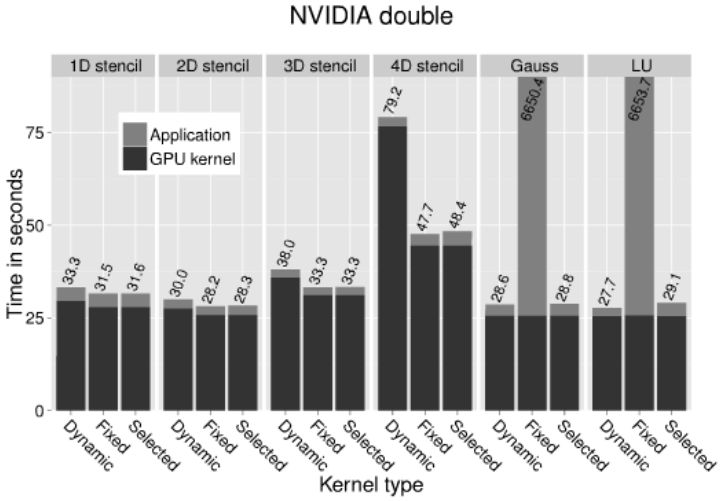
GPU Specs

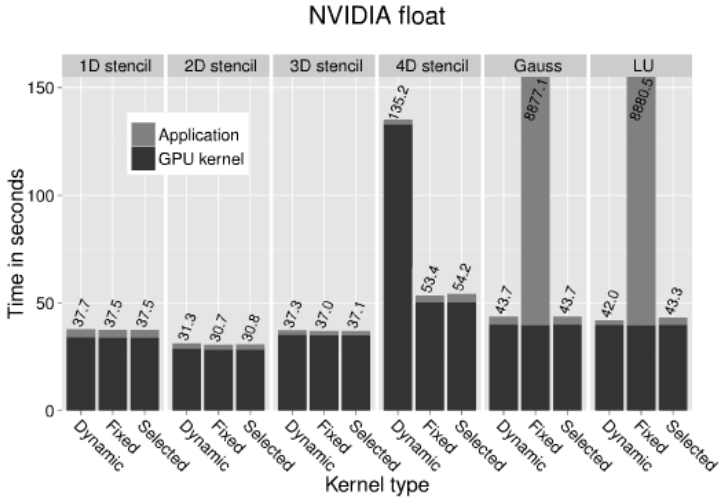
Vendor:	AMD	NVIDIA
Model:	HD 7970	GTX 680
Driver version:	1214.3 (VM)	331.38
#Cores:	2048	1536
Clock:	1000 MHz	1006 MHz
Memory:	3GB GDDR5	2GB DDR5
Memory bandwidth:	288 GB/s	192 GB/s
Peak performance:	4096 GFLOPS	3090 GFLOPS

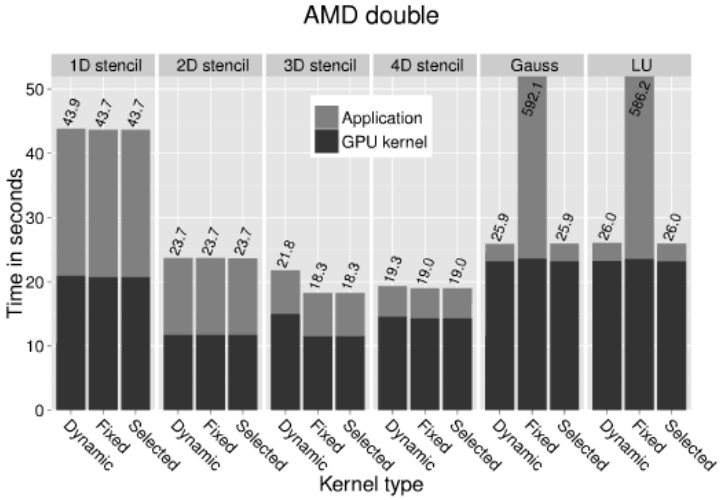
Compile times

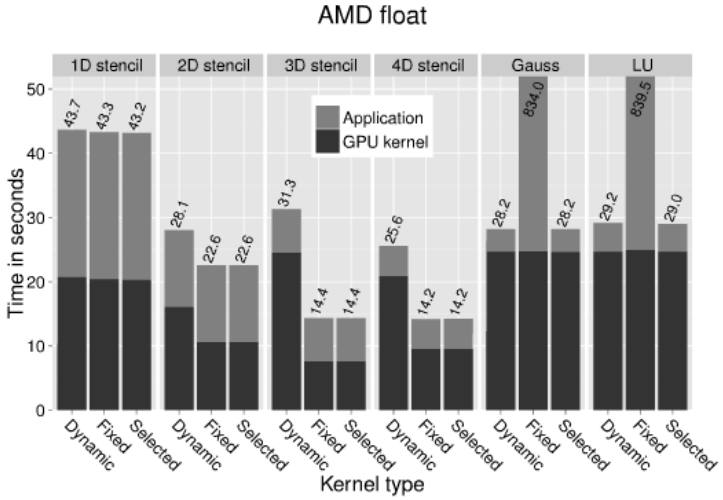
	NVIDIA	AMD	**ratio**
1D stencil	1445ms	117ms	**12.3**
2D stencil	1460ms	125ms	**11.7**
3D stencil	1503ms	151ms	**9.9**
4D stencil	3993ms	482ms	**8.3**
Gauss	2199ms	186ms	**11.8**
LU	2929ms	241ms	**12.2**
Average			***10.4***

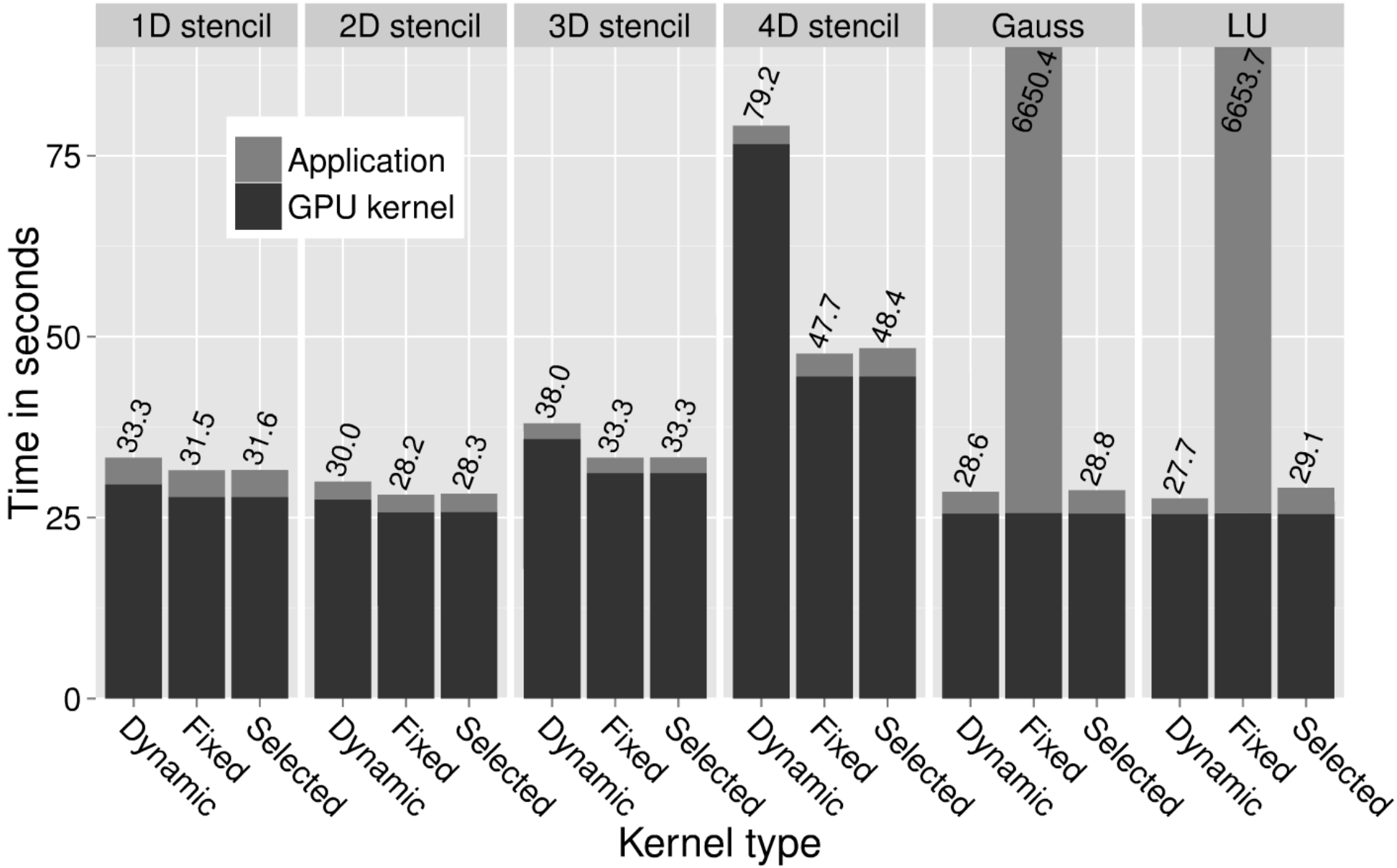
Benchmarks

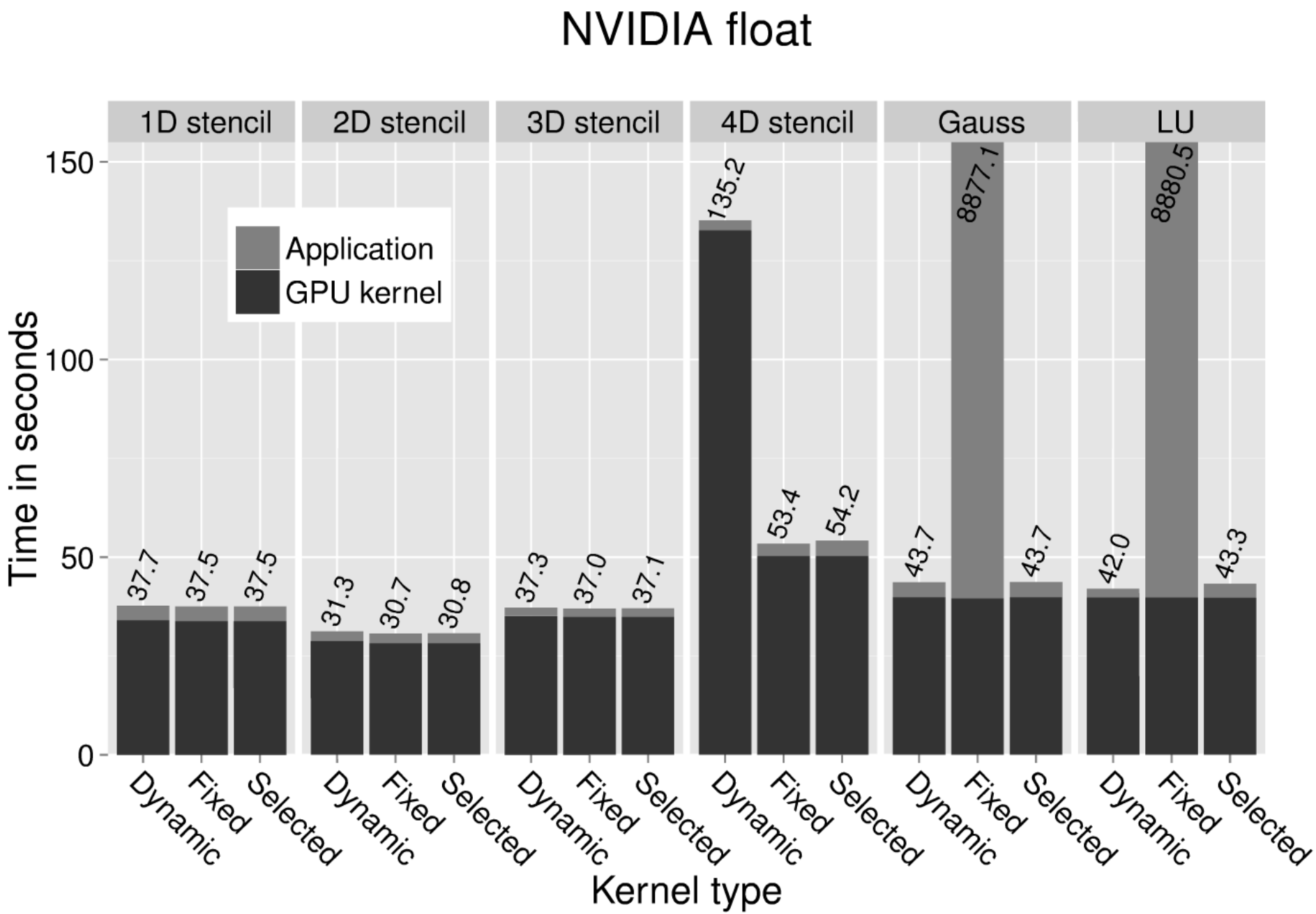

NVIDIA double
1D stencil
2D stencil
3D stencil
4D stencil
Gauss
LU
Application
GPU kernel
Time in seconds
Kernel type
Dynamic
Fixed
Selected
33.3
31.5
31.6
30.0
28.2
28.3
38.0
33.3
33.3
79.2
47.7
48.4
28.6
6650.4
28.8
27.7
6653.7
29.1

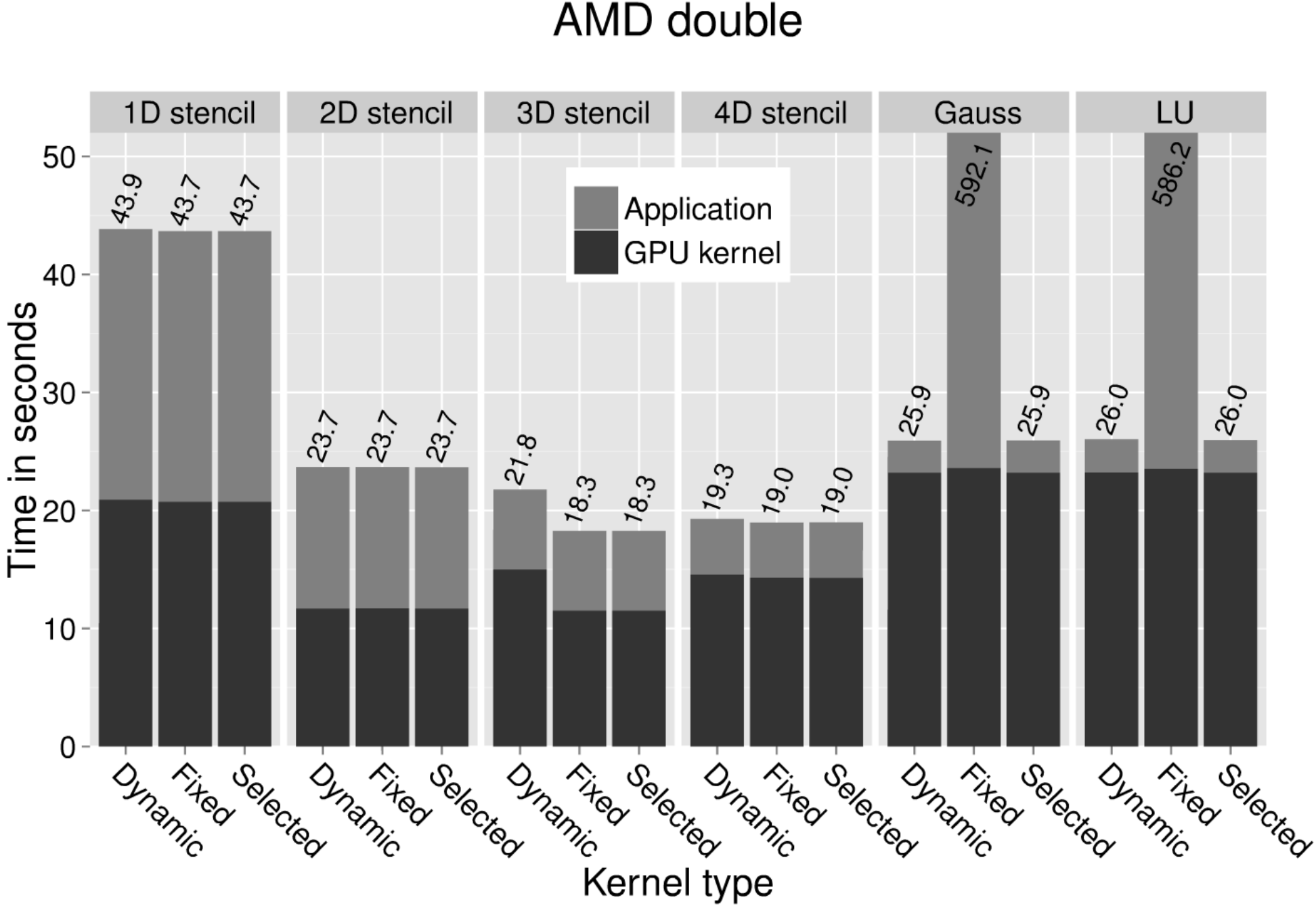

NVIDIA float
1D stencil
2D stencil
3D stencil
4D stencil
Gauss
LU
Application
GPU kernel
Time in seconds
Kernel type
Dynamic
Fixed
Selected
37.7
37.5
37.5
31.3
30.7
30.8
37.3
37.0
37.1
135.2
53.4
54.2
43.7
8677.1
43.7
42.0
8680.5
43.3

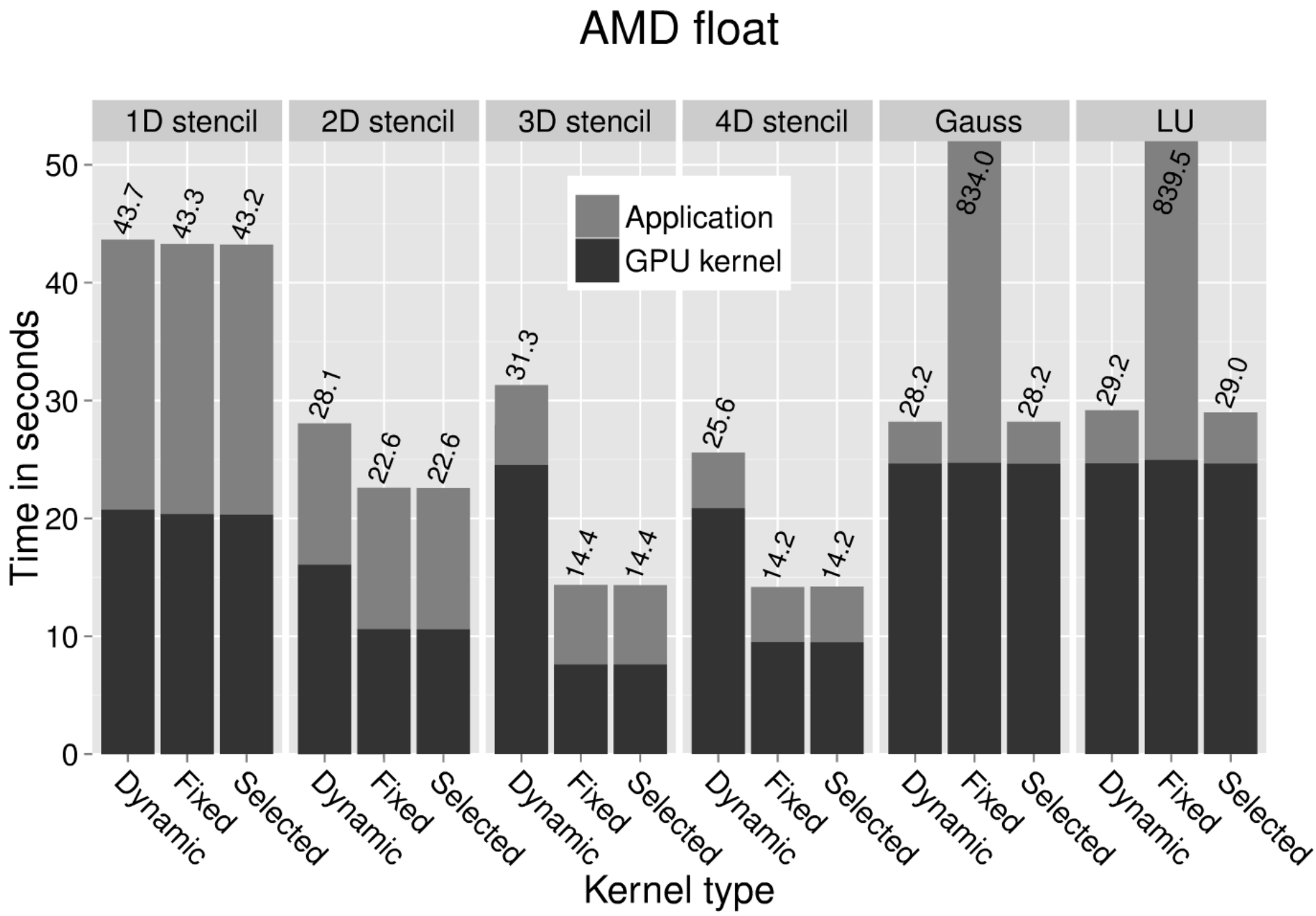

AMD double
1D stencil
2D stencil
3D stencil
4D stencil
Gauss
LU
Application
GPU kernel
Time in seconds
Kernel type
Dynamic
Fixed
Selected
43.9
43.7
43.7
23.7
23.7
23.7
21.8
18.3
18.3
19.3
19.0
19.0
25.9
592.1
25.9
26.0
566.2
26.0


AMD float
1D stencil
2D stencil
3D stencil
4D stencil
Gauss
LU
Application
GPU kernel
Time in seconds
Kernel type
Dynamic
Fixed
Selected
43.7
43.3
43.2
28.1
22.6
22.6
31.3
14.4
14.4
25.6
14.2
14.2
28.2
834.0
28.2
29.2
839.5
29.0


NVIDIA double
1D stencil
2D stencil
3D stencil
4D stencil
Gauss
LU
Application
GPU kernel
Time in seconds
0
25
50
75
33.3
31.5
31.6
30.0
28.2
28.3
38.0
33.3
33.3
79.2
47.7
48.4
28.6
6650.4
28.8
27.7
6653.7
29.1
Dynamic
Fixed
Selected
Kernel type


NVIDIA float
1D stencil
2D stencil
3D stencil
4D stencil
Gauss
LU
Application
GPU kernel
Time in seconds
150
100
50
0
37.7
37.5
37.5
31.3
30.7
30.8
37.3
37.0
37.1
135.2
53.4
54.2
43.7
8877.1
43.7
42.0
8880.5
43.3
Dynamic
Fixed
Selected
Kernel type


AMD double
1D stencil
2D stencil
3D stencil
4D stencil
Gauss
LU
Application
GPU kernel
Time in seconds
0
10
20
30
40
50
43.9
43.7
43.7
23.7
23.7
23.7
21.8
18.3
18.3
19.3
19.0
19.0
25.9
592.1
25.9
26.0
586.2
26.0
Dynamic
Fixed
Selected
Kernel type


AMD float
1D stencil
2D stencil
3D stencil
4D stencil
Gauss
LU
Application
GPU kernel
Time in seconds
0
10
20
30
40
50
43.7
43.3
43.2
28.1
22.6
22.6
31.3
14.4
14.4
25.6
14.2
14.2
28.2
834.0
28.2
29.2
839.5
29.0
Dynamic
Fixed
Selected
Kernel type

Thank You

http://bh107.org