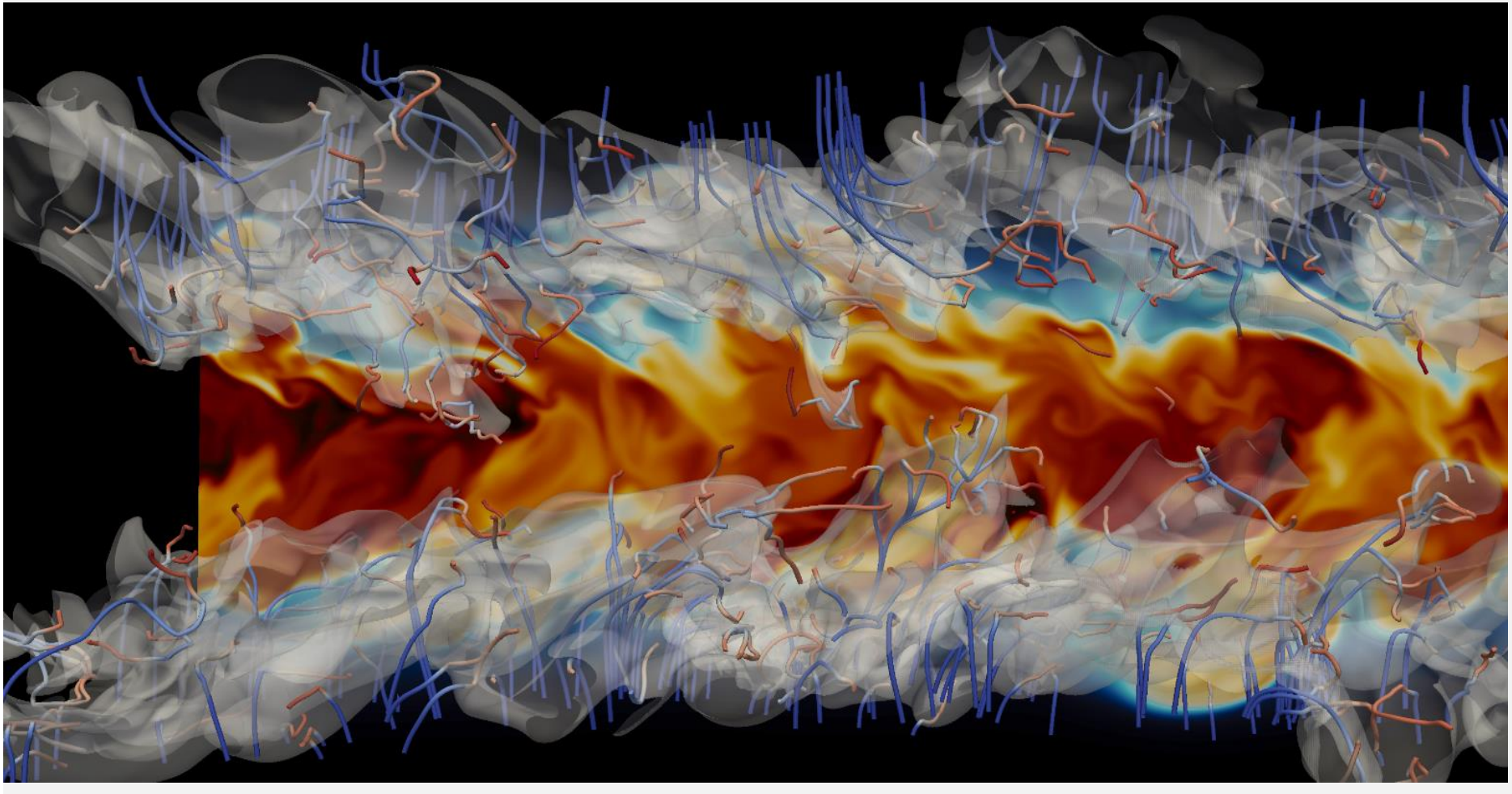
Software Tools for UNIX/Linux Systems

Part 7: Optimization

C. Hasse

TECHNISCHE UNIVERSITÄT DARMSTADT



Optimization

- Modern compiler can optimize the code
- A wide range of optimization are possible

Example

```
double a[2] = {0.3, 0.4}
a[1] = a[1]+1.5
```

Address calculated twice

OPTIMIZATION

```
double a[2] = {0.3, 0.4}
double* tmp = a + 1
*tmp += 1.5
```

Address calculated once

Optimization

- Many algorithm and methodology available for optimization
  - Code hoisting
  - Constant propagation
  - Common subexpression elimination
  - Dead code elimination
  - Operator optimization
  - and many some….

OPERATOR OPTIMIZATION

1.465e+5/10

▼

1.465e+(5-1)

- Can heavily improve code performance
- Optimization level can be tuned
  - Higher level → More optimization algorithm enabled

GCC syntax for optimization

`gcc -Ox [input]`	x is an integer from 0 to 3 0 = lowest level 3 = highest level

Optimization

- Check the source code optimization.c
  - Try to understand why is no a good code
- Compile the code optimization.c with different optimization level
- Measure the execution time

```
$> gcc –O0 optimization.c
$> time ./a.out
$> gcc –O3 optimization.c
$> time ./a.out
```

- Optimization is not linear with the level

Optimization

- Optimization does NOT dispense with write good code!
- Remember: compiler is a program, it is very advance but it does not know what you want
- Try to compile overflow.c with different optimization level
- Optimization can introduce unexpected behavior!

- Be aware of:
  - Overflow arithmetic
  - Pointer aliasing
  - Clearing memory