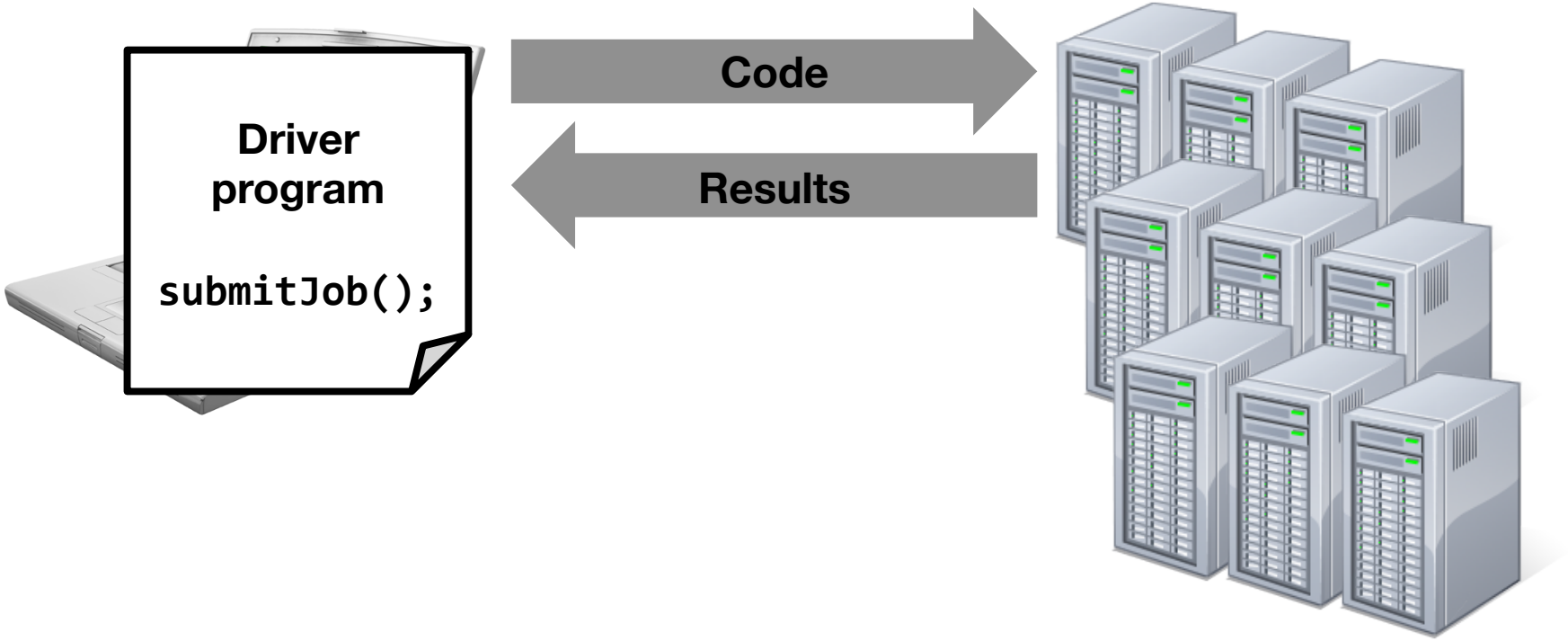
Programming the cloud with Skywriting

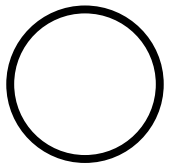
Derek G. Murray
Malte Schwarzkopf Chris Smowton
Anil Madhavapeddy Steven Hand

University of Cambridge

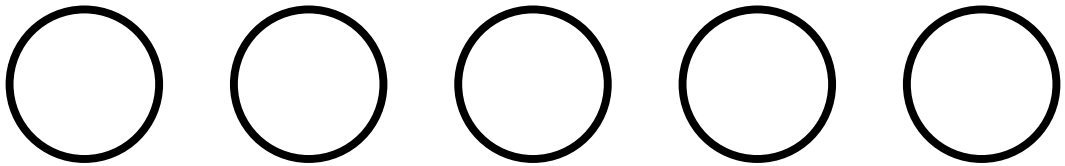
Move computation to the data



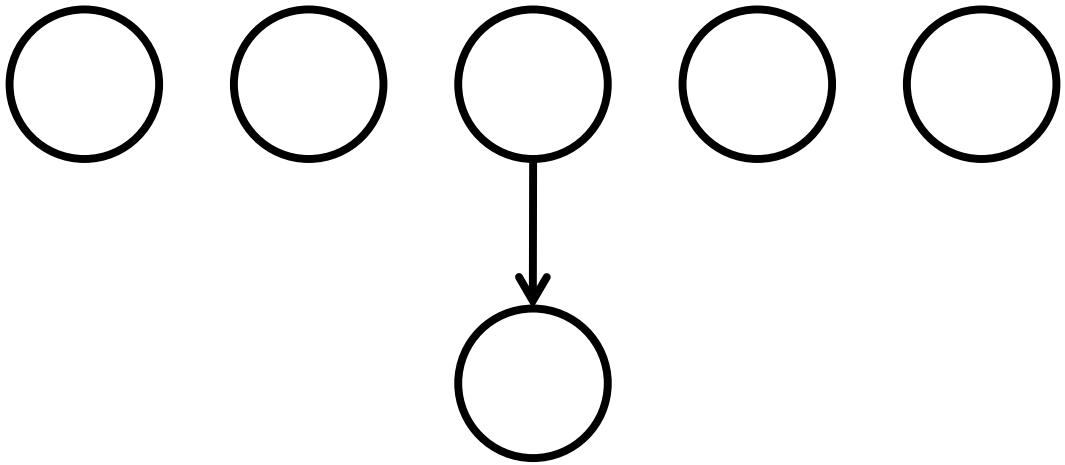
Task-based parallelism



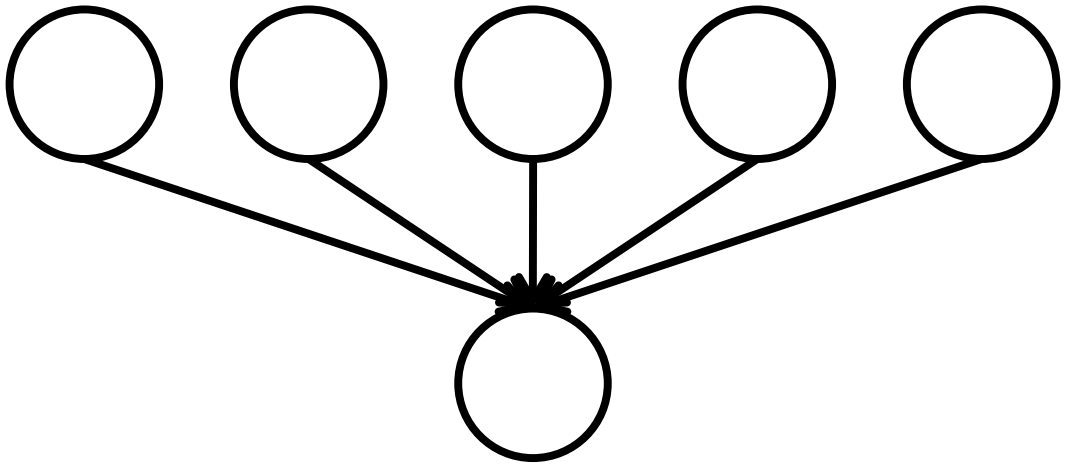
Independent tasks



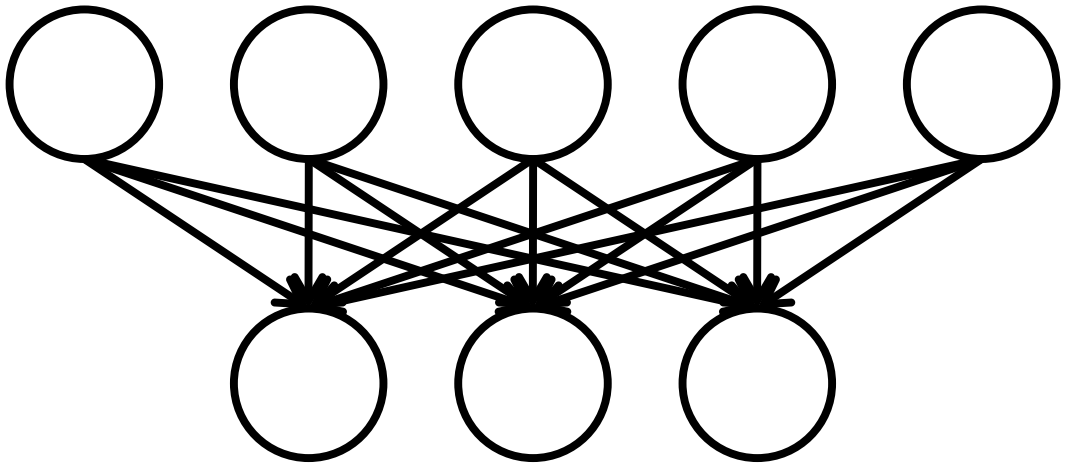
Task dependencies



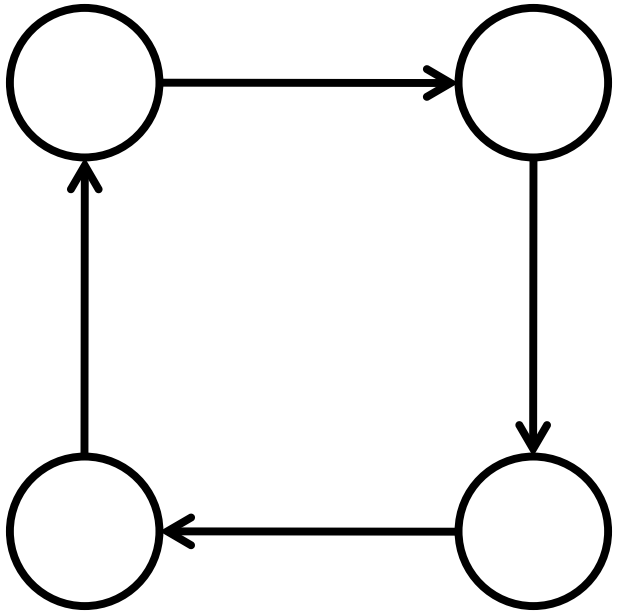
Task dependencies



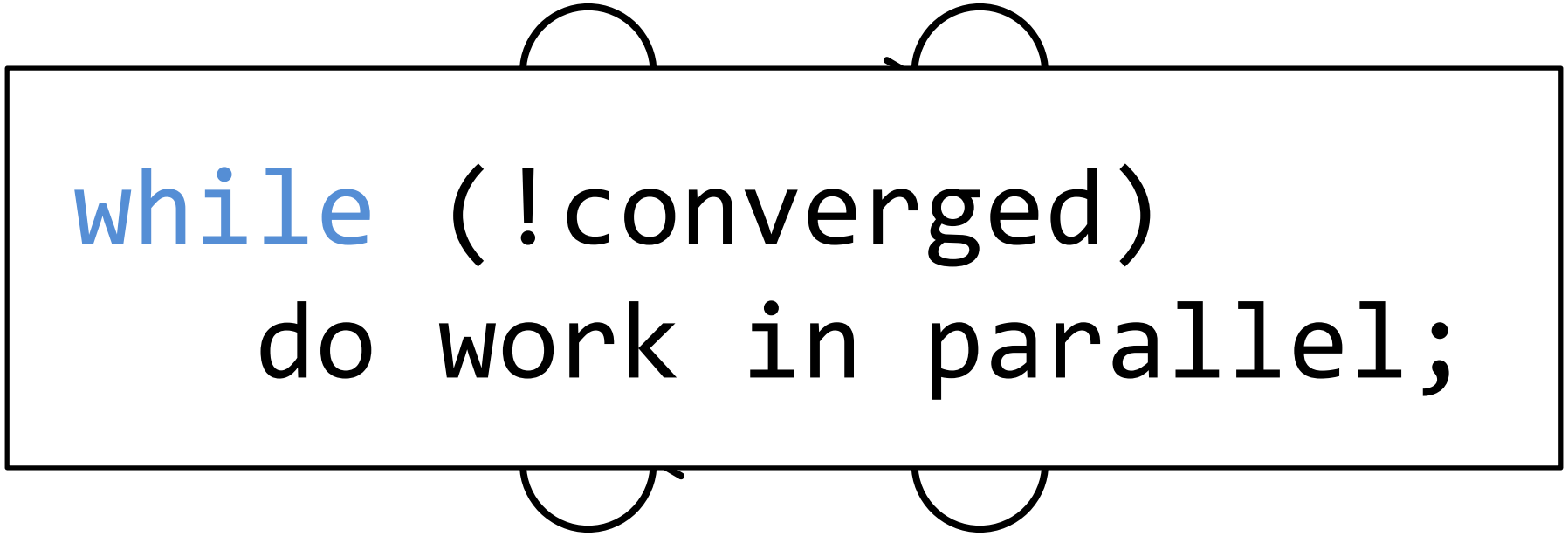
MapReduce



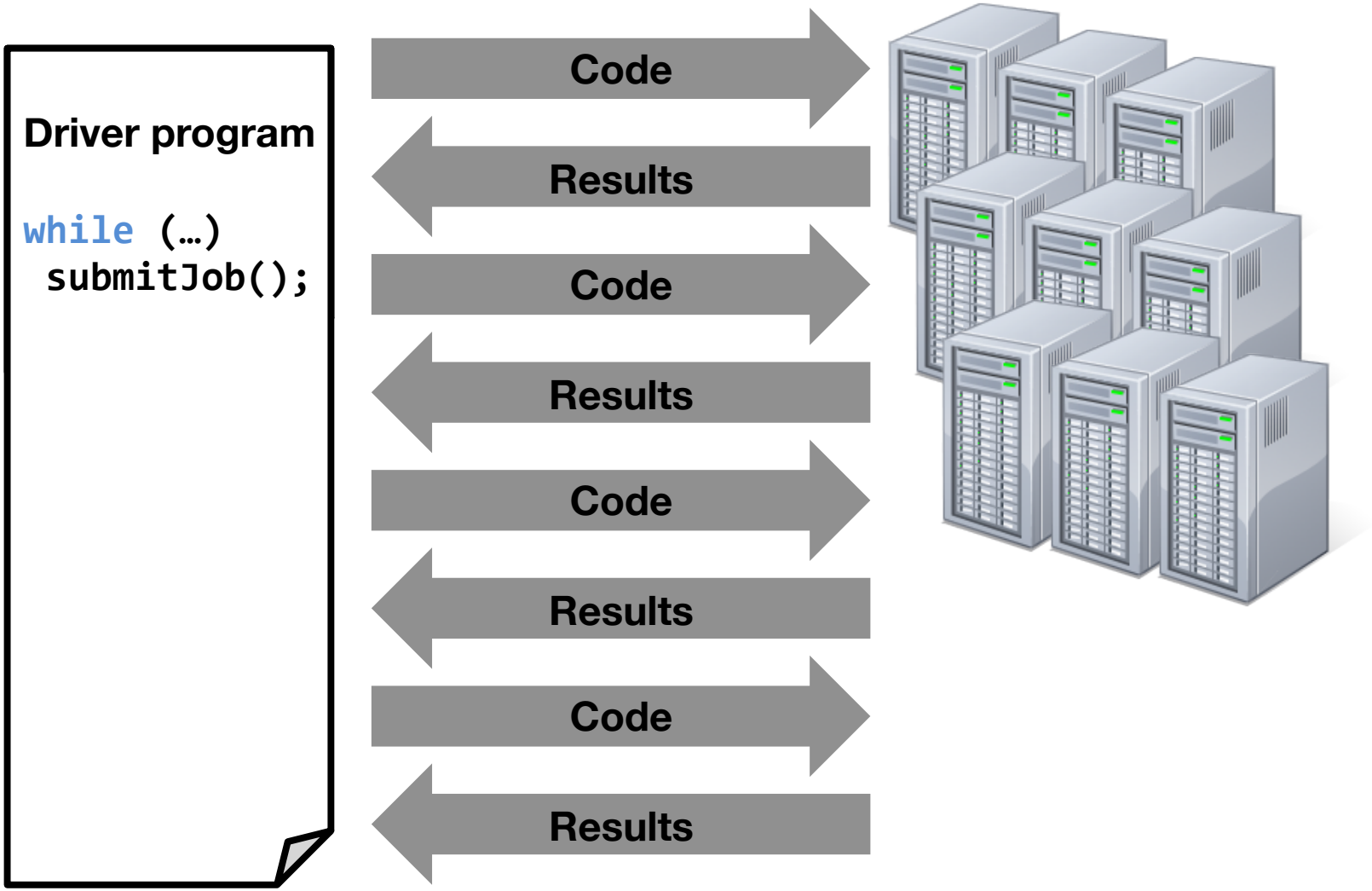
None of the above



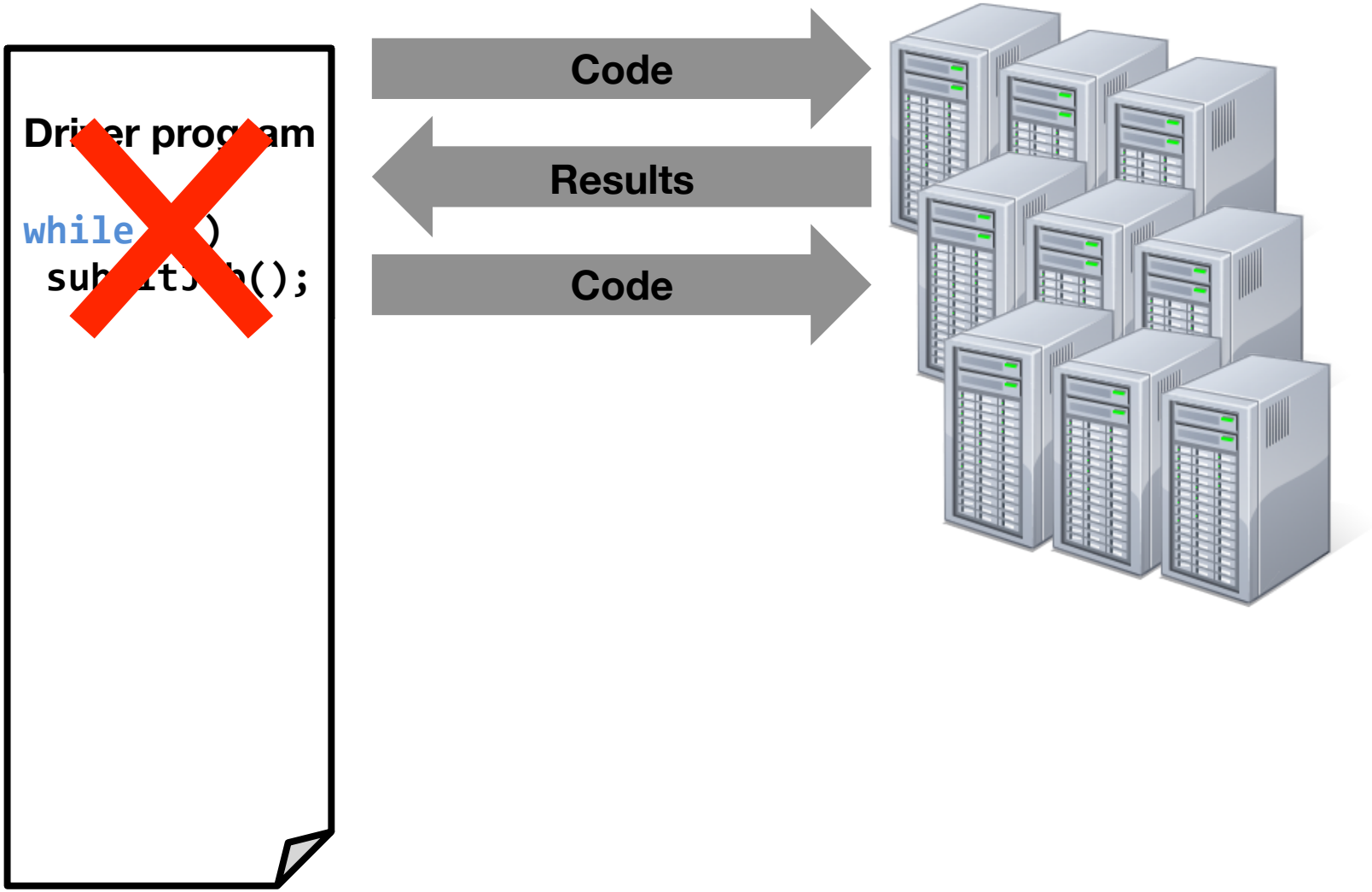
None of the above



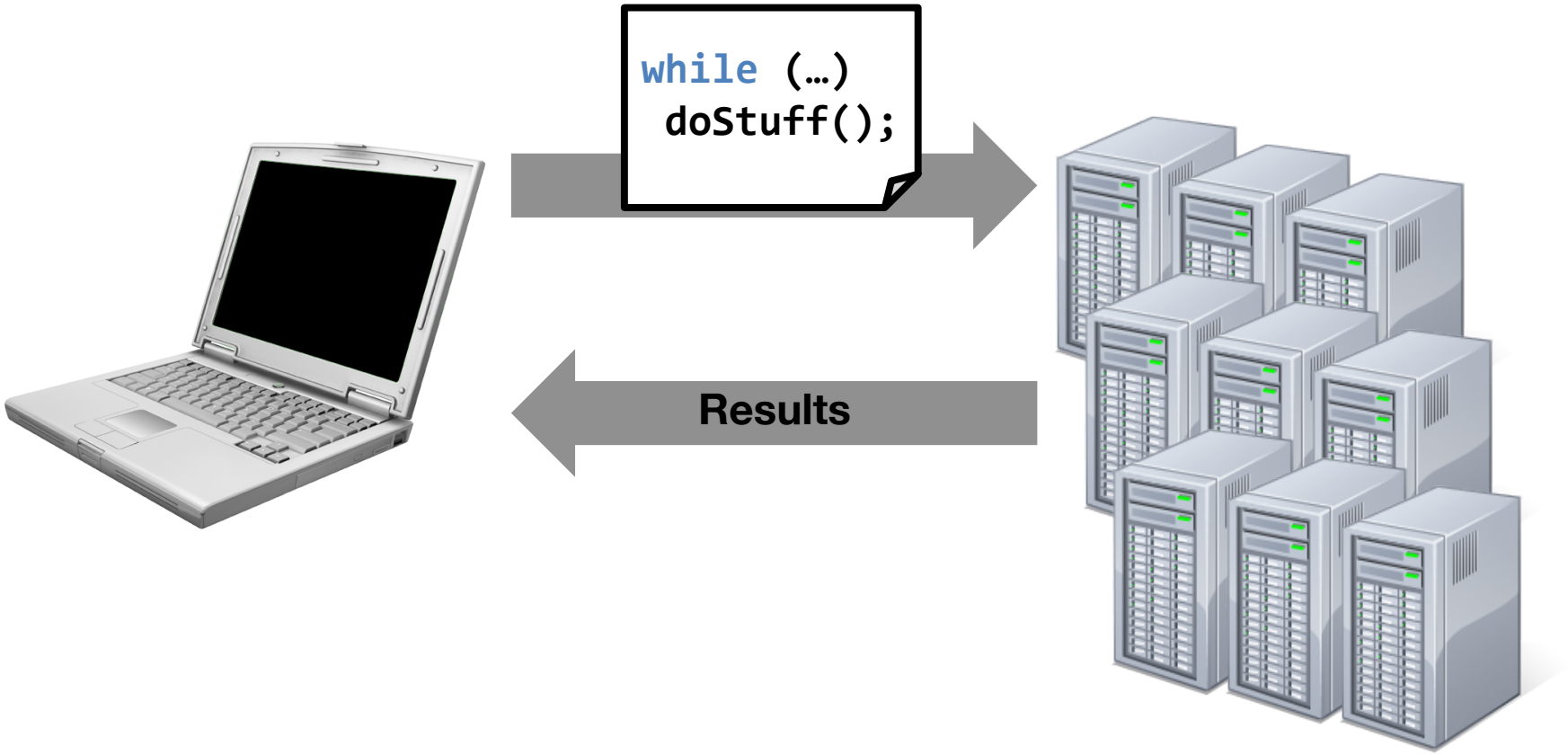
Iterative algorithm



Iterative algorithm



Skywriting



Skywriting

- JavaScript-like job specification language
  - Supports functional programming
  - Data-dependent control flow
- Distributed execution engine
  - Locality-based scheduling
  - Fault tolerance
  - Thread migration

Spawning a task

```
function f(x) { return x + 1; }


res1 = spawn(f, [42]);
```

Task dependencies

```
function f(x) { return x + 1; }
function g(y) { … }

res1 = spawn(f, [42]);
res2 = spawn(g, [res1]);
```

`res1` and `res2` are *future references*

PageRank

```
pages = […]; // List of partitions
x = …;       // Random initial value
do {
  x_old = x;
  results = [];
  for (part in pages) {
    results += spawn(pagerank, [part, x_old]);
  }
  x = spawn(collect, [results]);
  done = spawn(converged, [x_old, x]);
} while (!*done);
```

PageRank

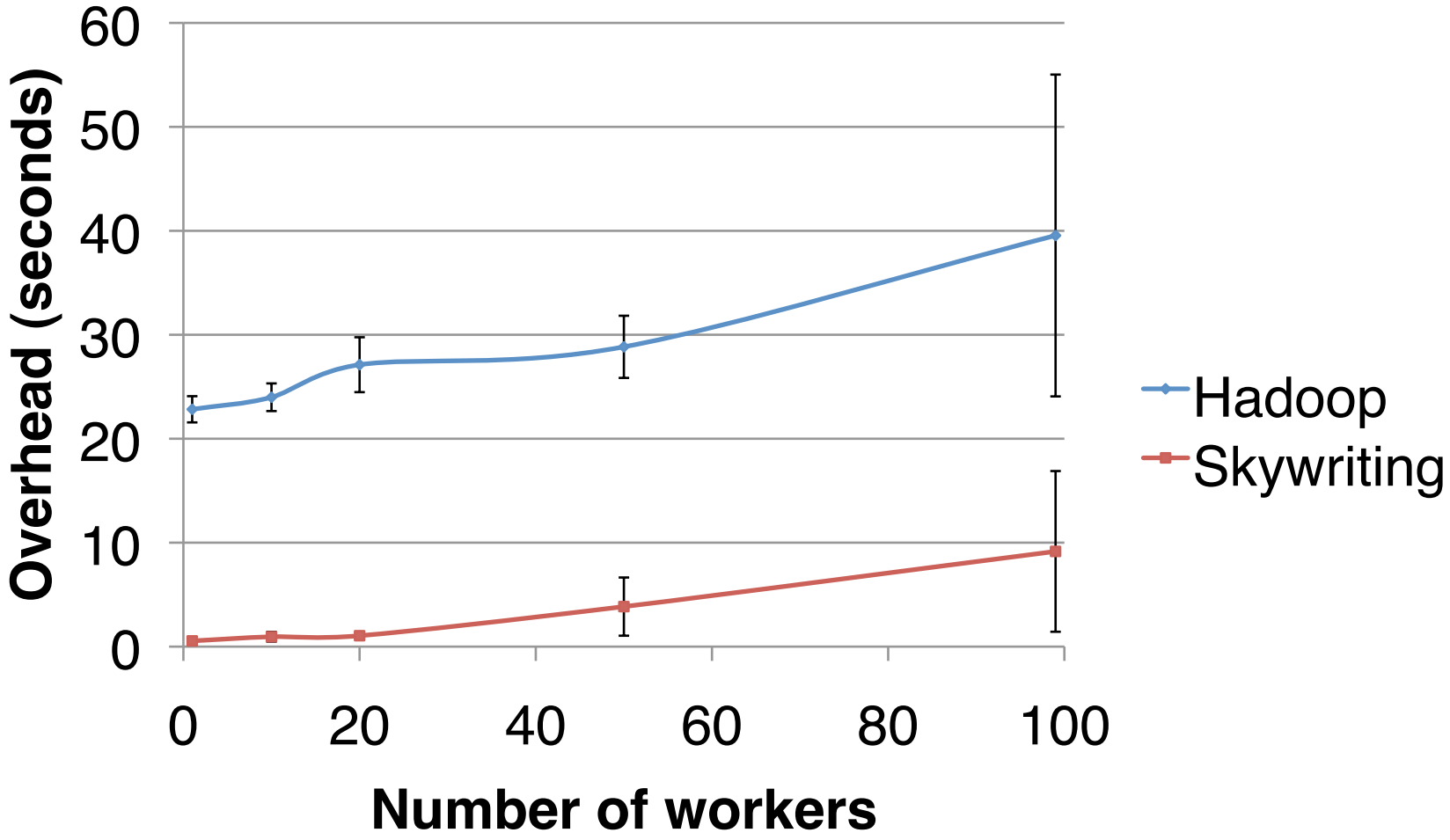
```
pages = […]; // List of partitions
x = …;       // Random initial value
do {
  x_old = x;
  results = [];
  for (part in pages) {
    results += spawn(pagerank, [part, x_old]);
  }
  done = spawn(converged, [x_old, x]);
} while (!*done);
```

*-operator dereferences (forces) a future

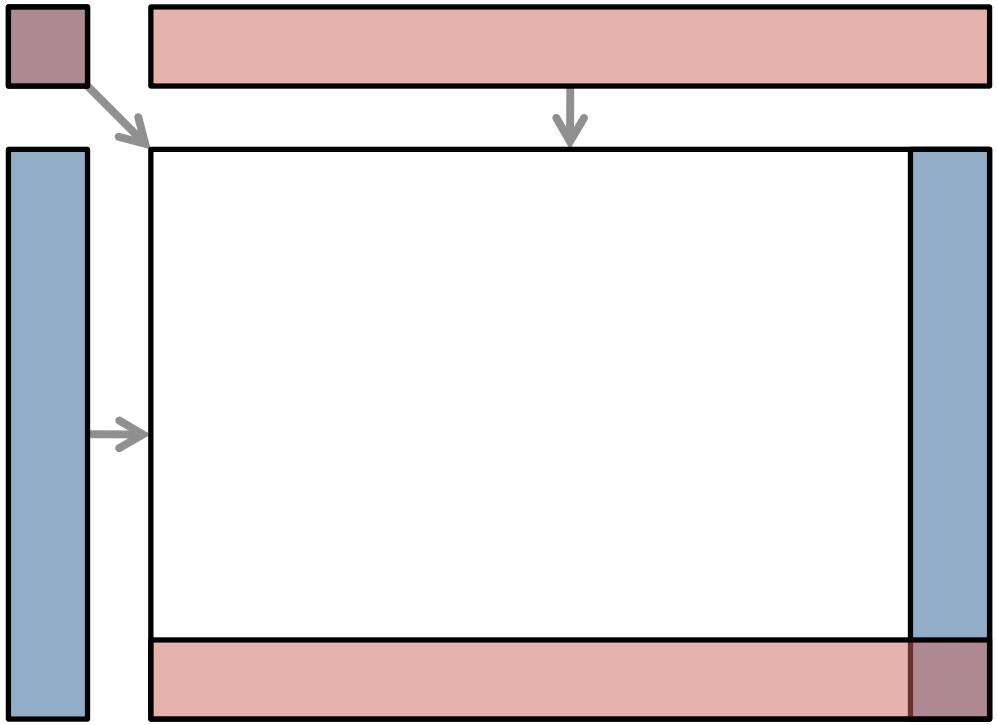
Implementation status

- Implemented in 4000 lines of Python
  - Also: Java, C and .NET bindings
- Many additional features
  - Native code execution
  - Introspection
  - Conditional synchronisation
- Available as open-source
  - http://github.com/mrry/skywriting

Job creation overhead

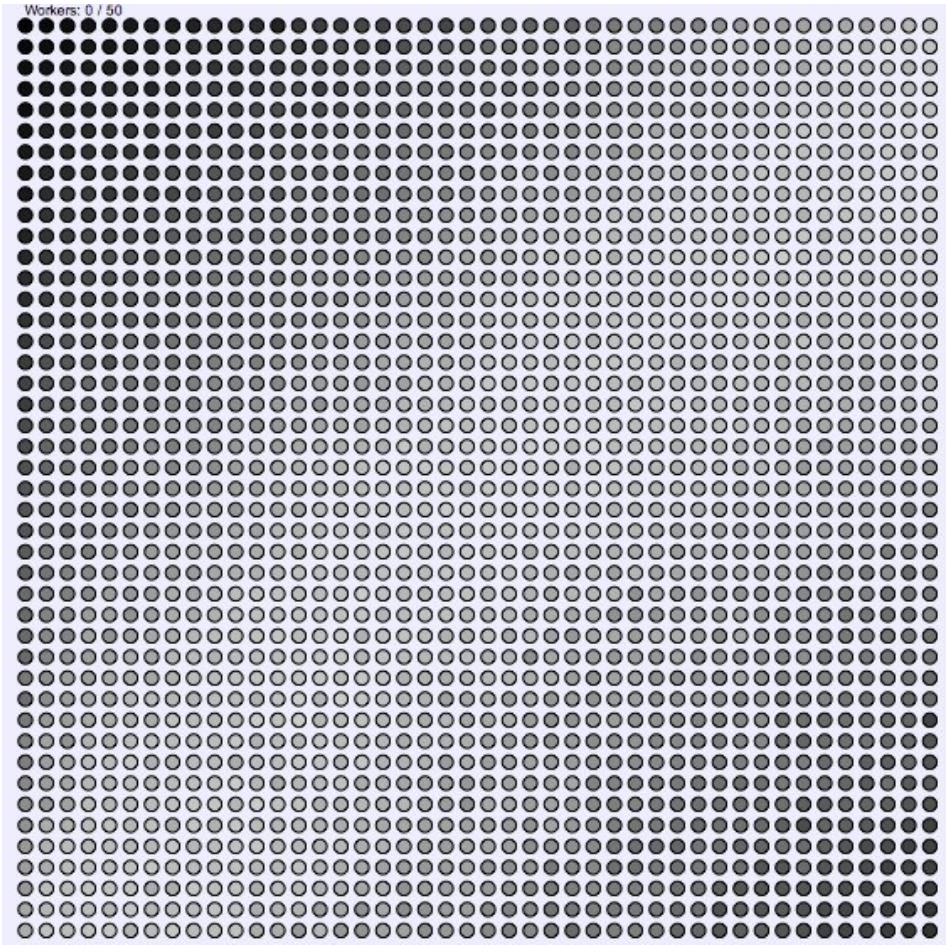


Smith-Waterman data flow

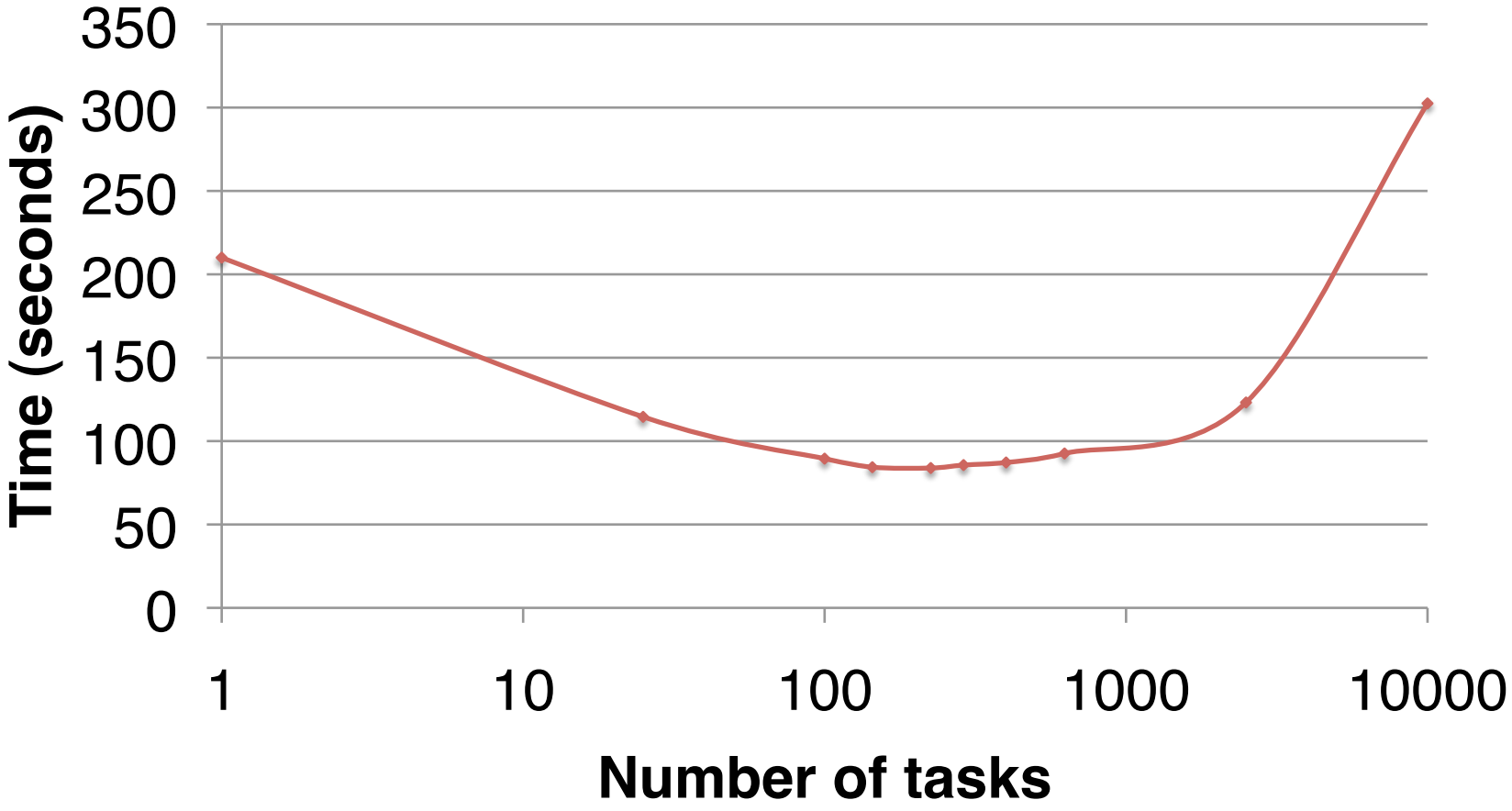


$$H_{i,j} = f\left(H_{i-1,j}, H_{i,j-1}, H_{i-1,j-1}, s_i, t_j\right)$$

Parallel Smith-Waterman



Parallel Smith-Waterman



Future directions

- Multiple-scale parallel computing
  - Multiple cores, machines and clouds
- Streaming computations
  - Piping high-bandwidth data between tasks
- Better language integration
  - Hosted Skywriting on CLR or JVM

Conclusions

- Turing-complete programming language for distributed computation
- Runs real jobs with low overhead
- Lots more still to do!

Questions?

- Email
  - Derek.Murray@cl.cam.ac.uk
- Project website
  - http://www.cl.cam.ac.uk/netos/skywriting/