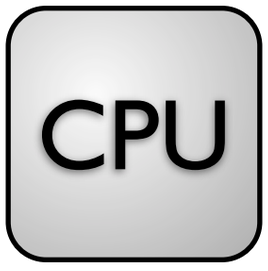
The Challenges of Irregular Parallelism

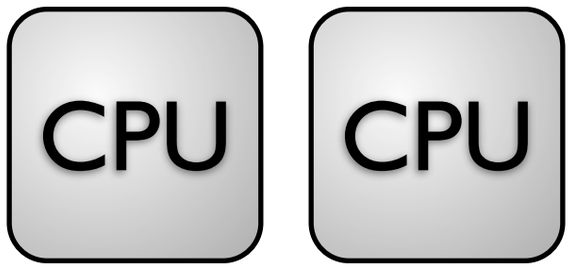
Chris Seaton
seatonc@cs.man.ac.uk
chrisseaton.com

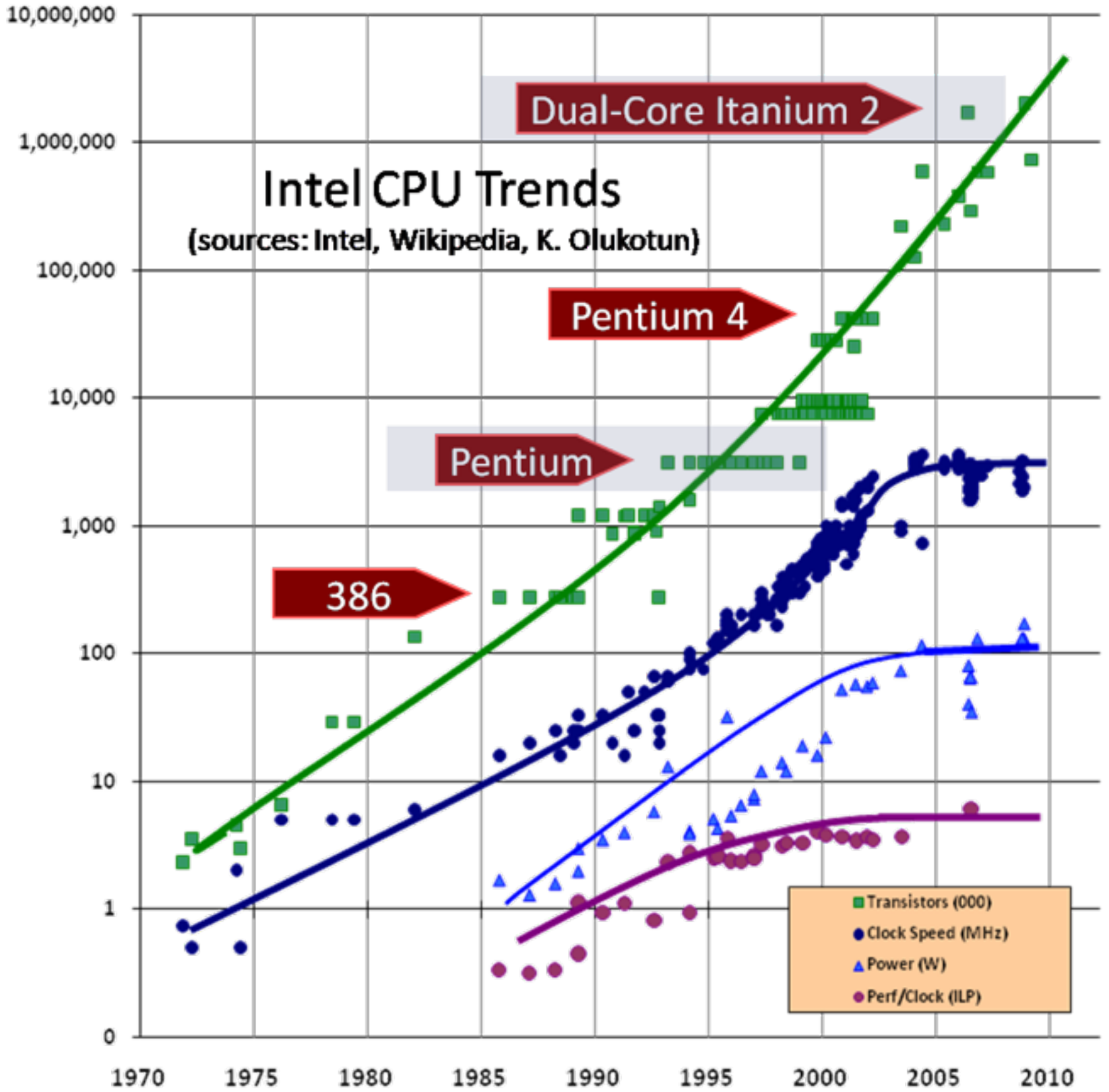
Advanced Processor Technologies Group

Supervisors: Ian Watson and Mikel Luján

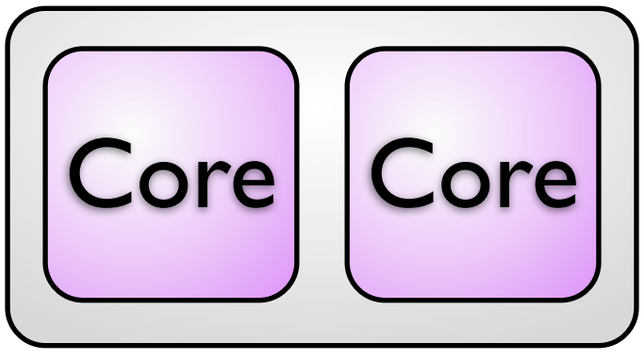
Why should I write parallel programs?

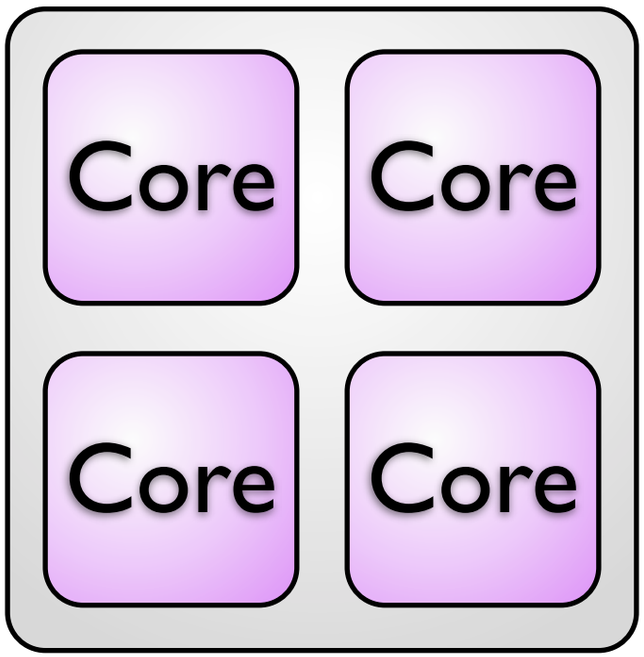

CPU

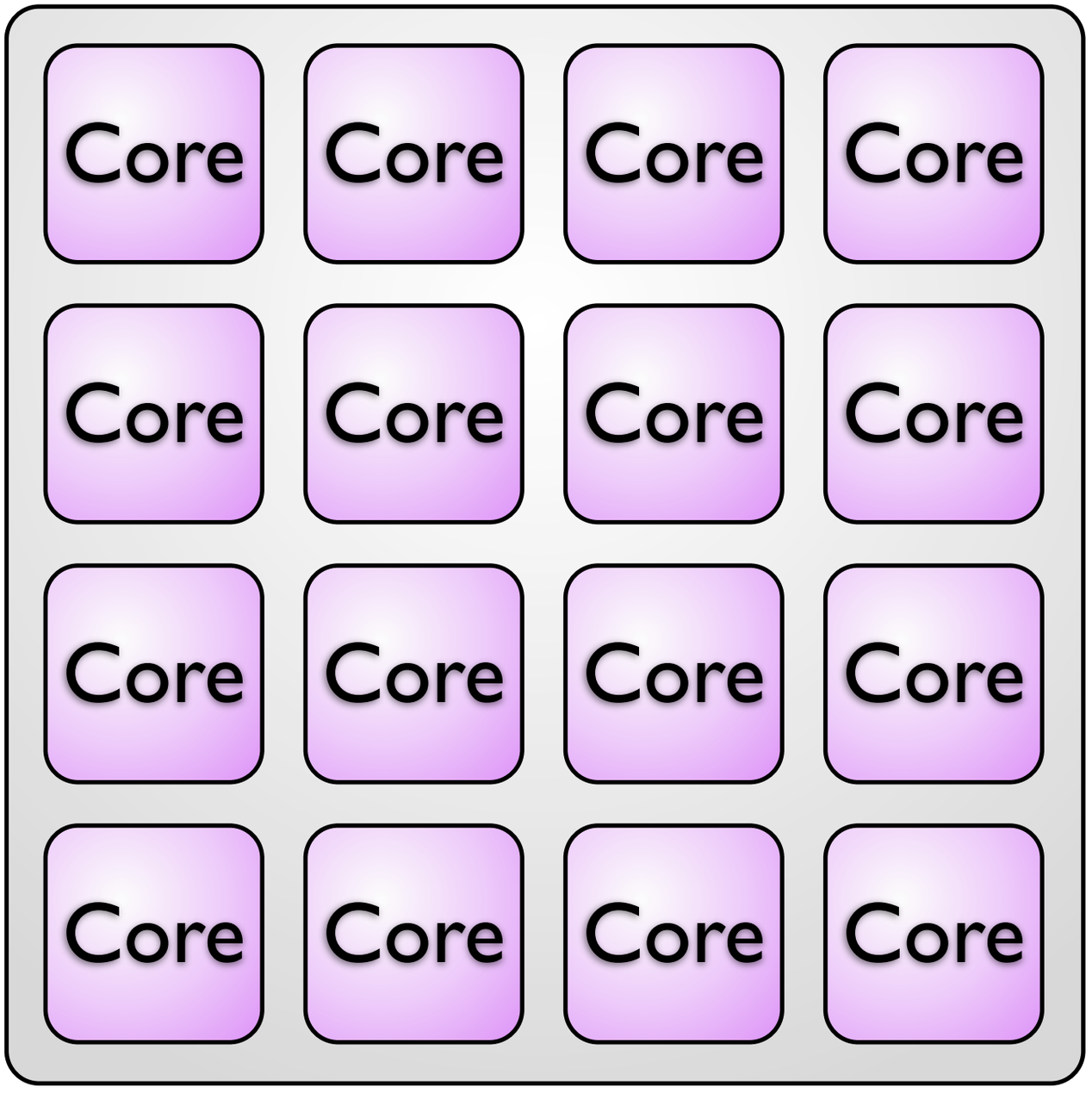

CPU
CPU

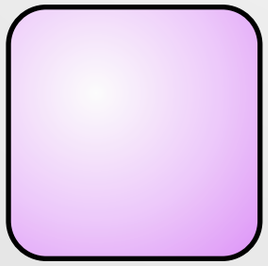


H. Stutter. The free lunch is over – a fundamental turn toward concurrency in software. Dr. Dobb's, 30(3), 2005.


Core
Core

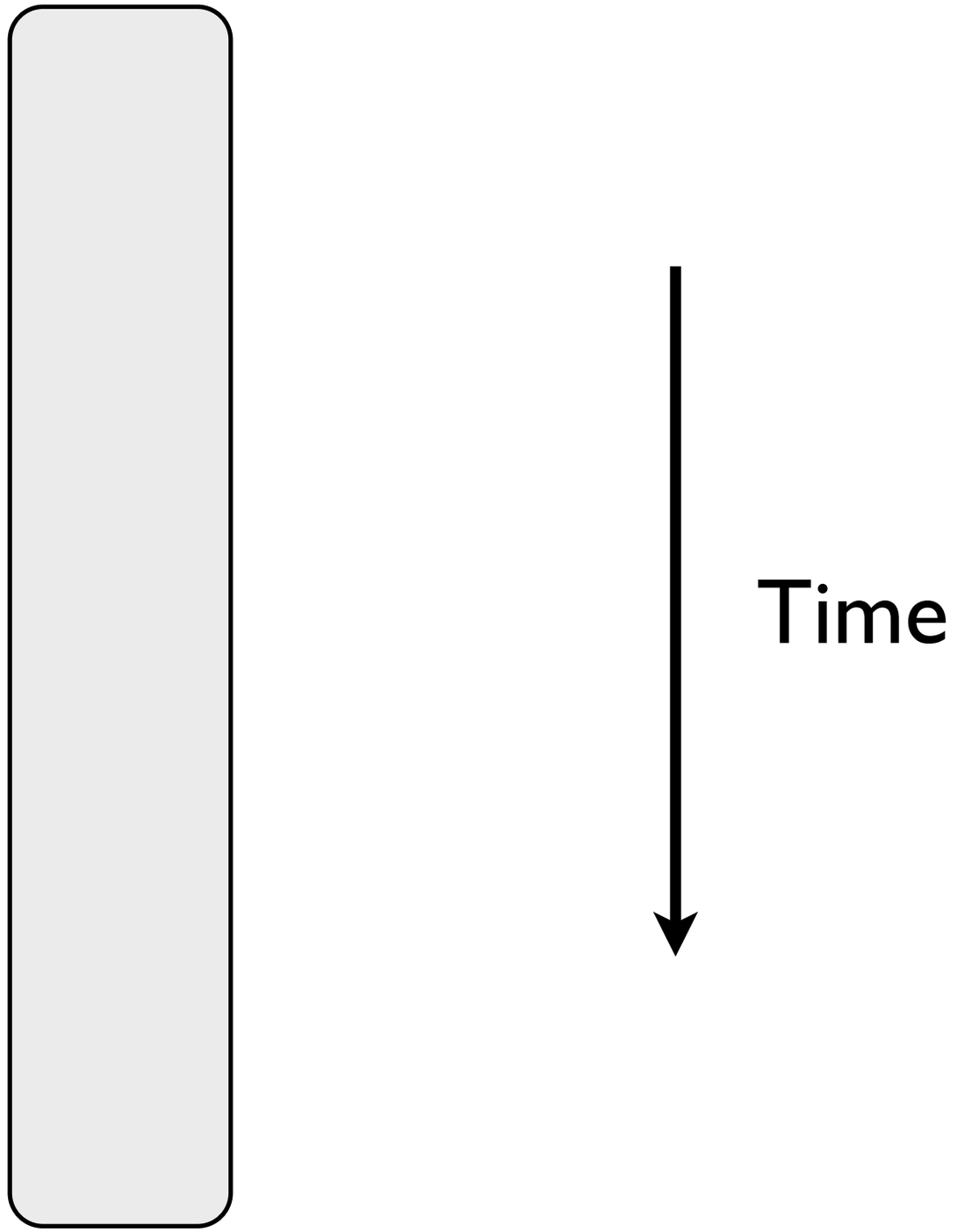

Core
Core
Core
Core

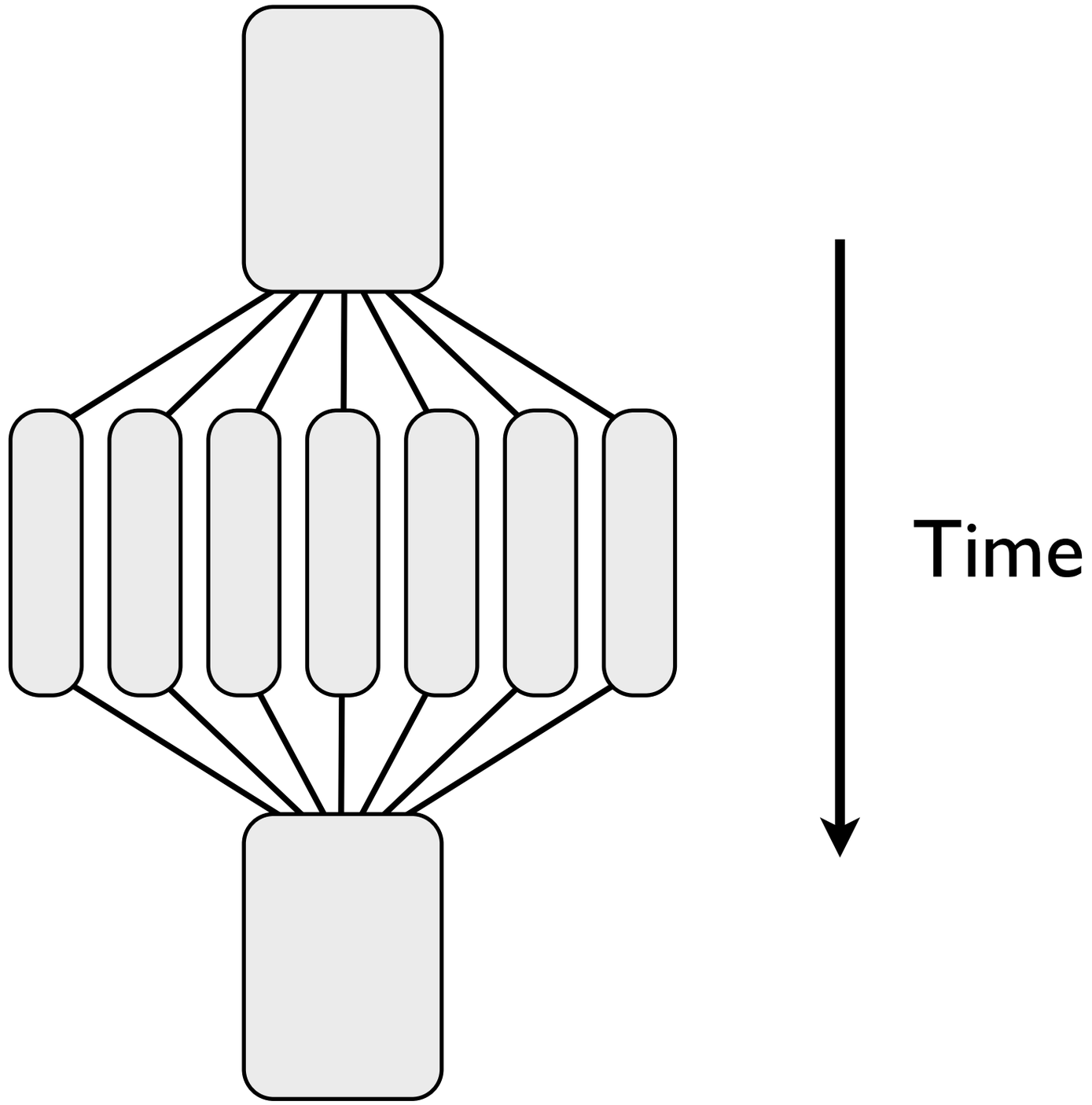

Core
Core
Core
Core
Core
Core
Core
Core
Core
Core
Core
Core
Core
Core
Core
Core



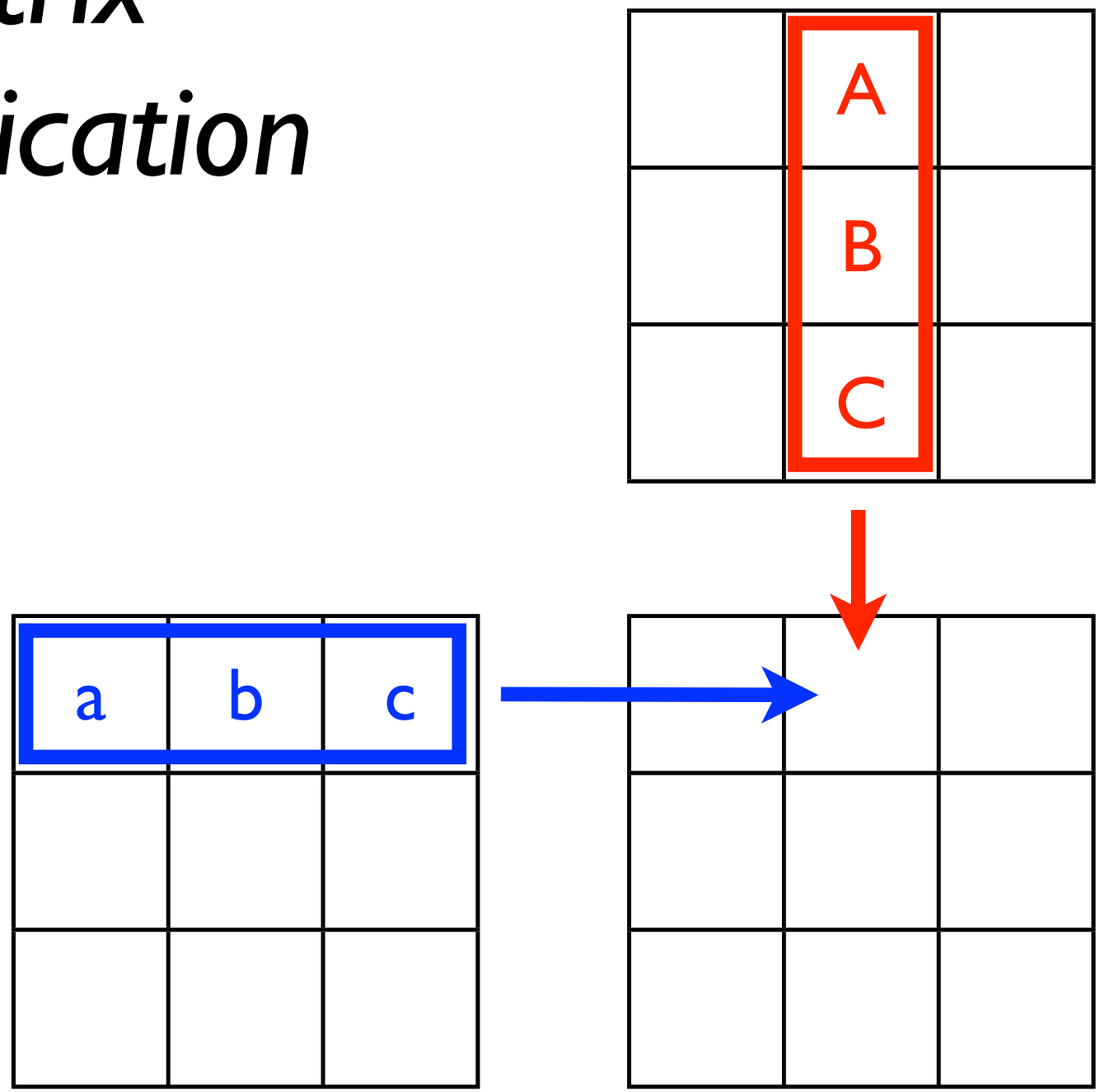
We already know how to make parallel programs

Structure your program so that there are tasks that can run at the same time


Time

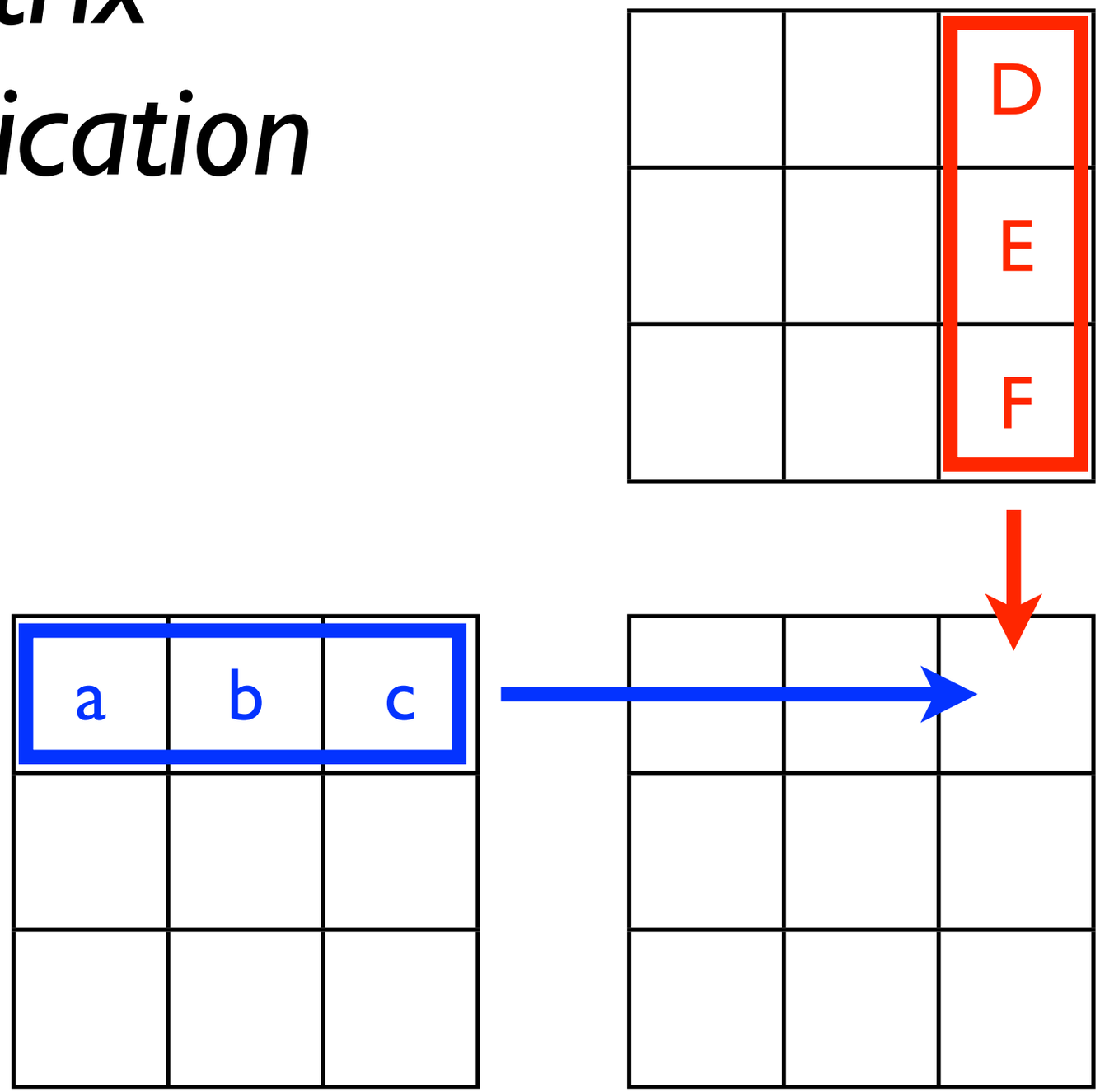

Time

Matrix multiplication



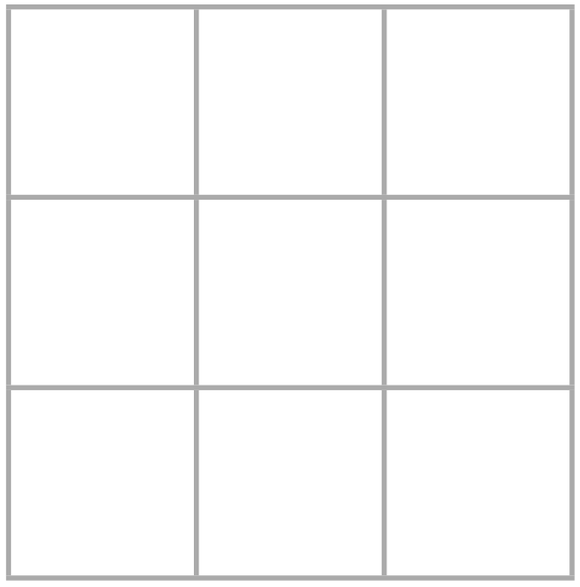
$$aA + bB + cC$$

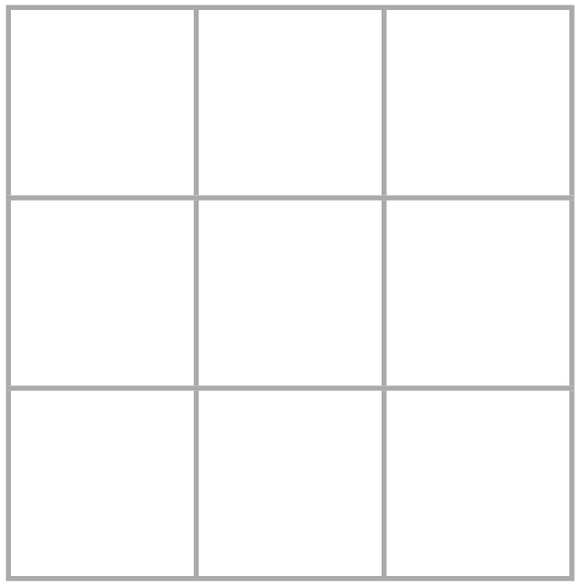
Matrix multiplication

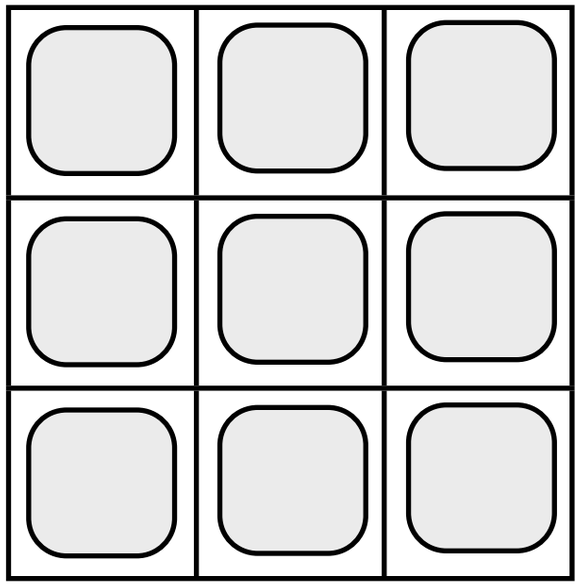


$$aD + bE + cF$$

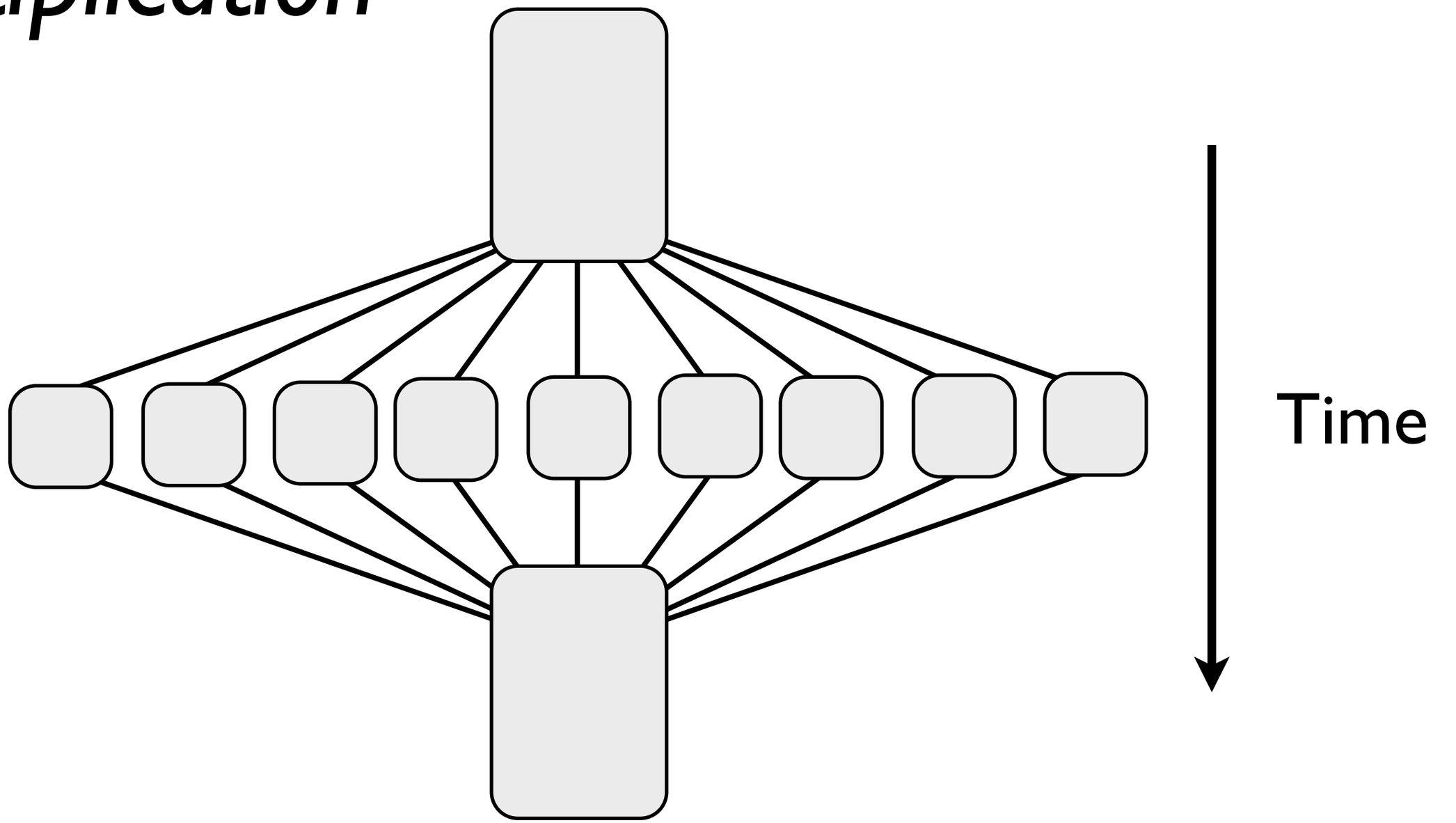
Matrix multiplication



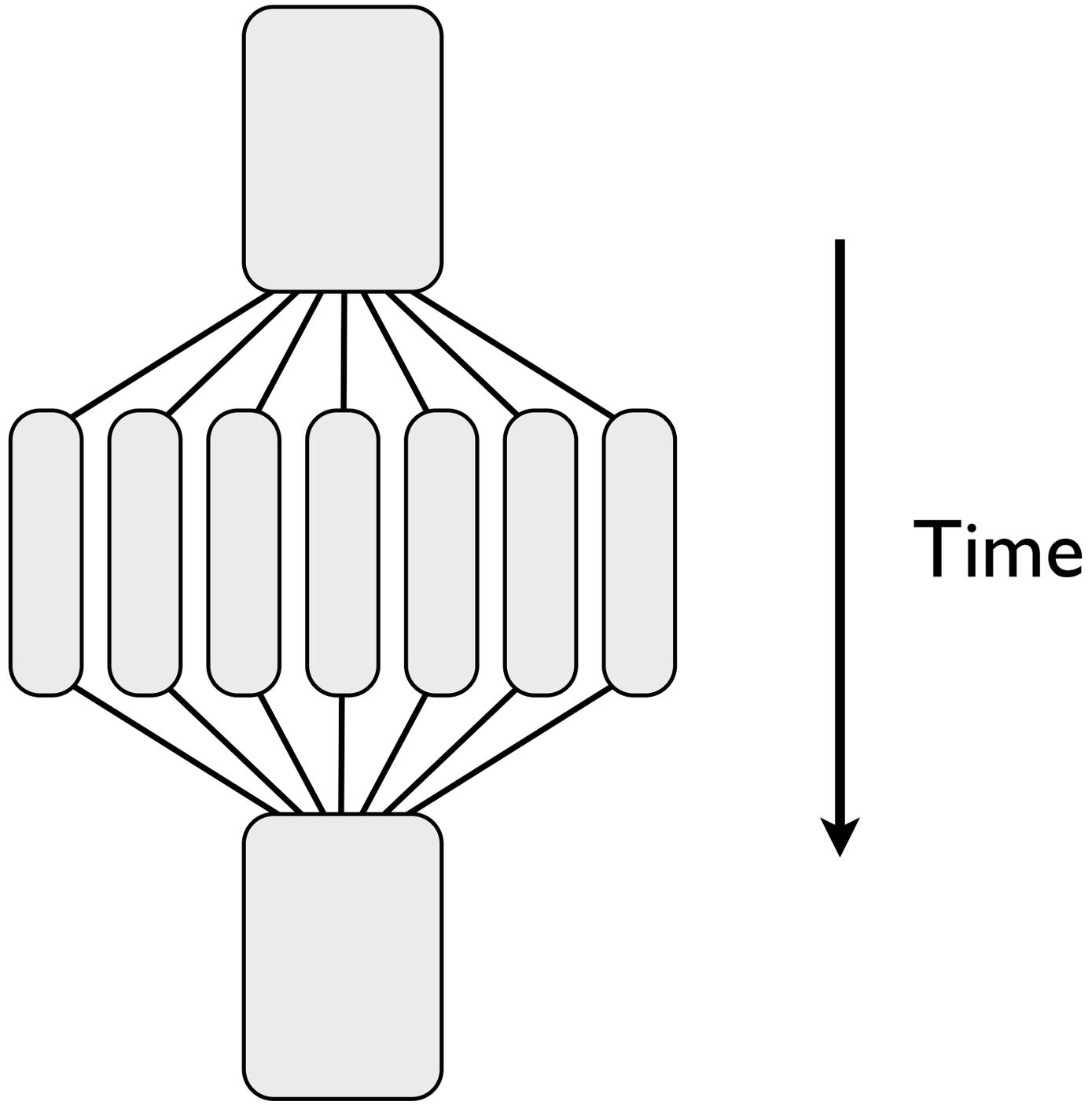




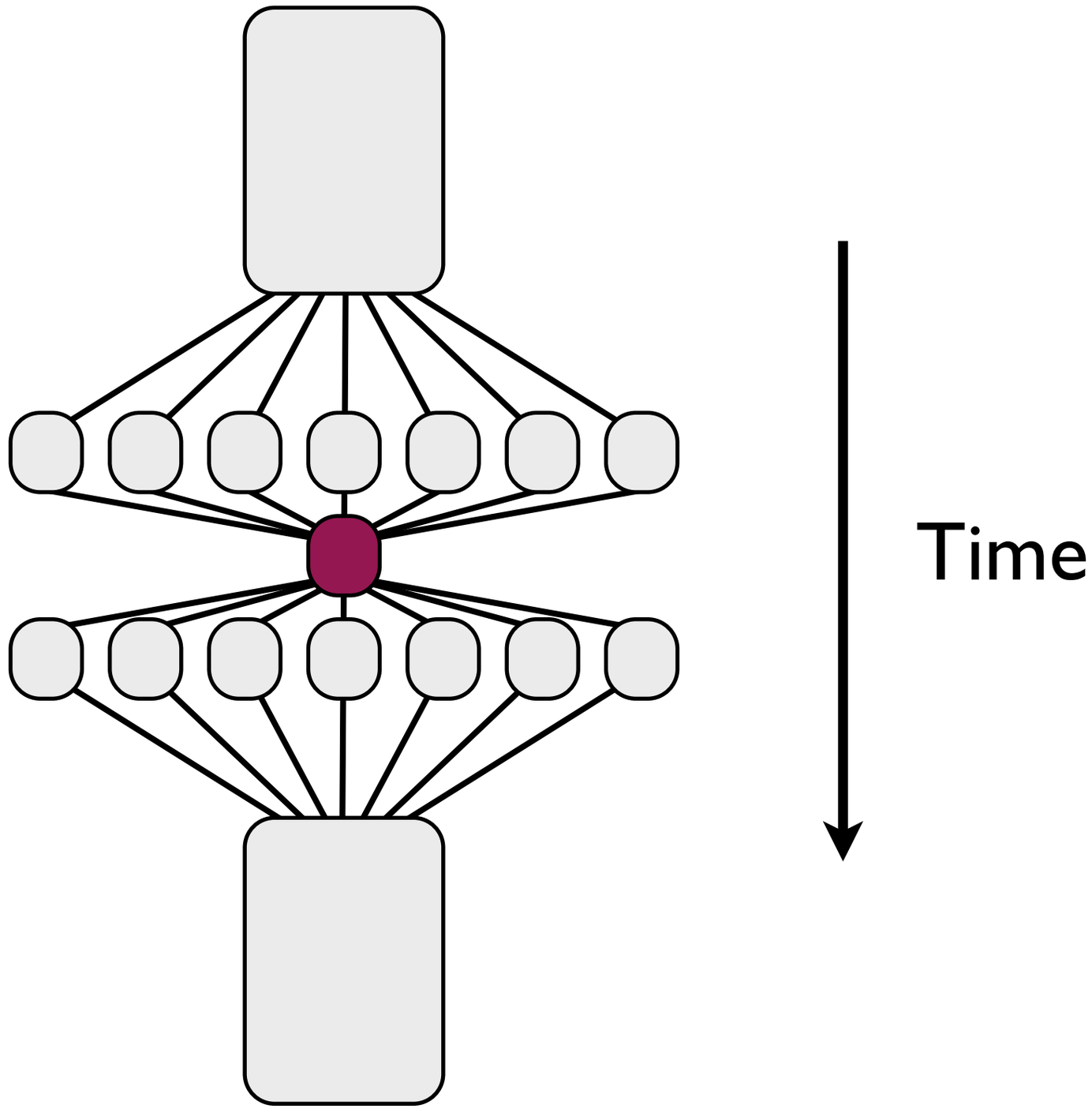
Matrix multiplication



However, this only works if those tasks are entirely independent of each other


Time

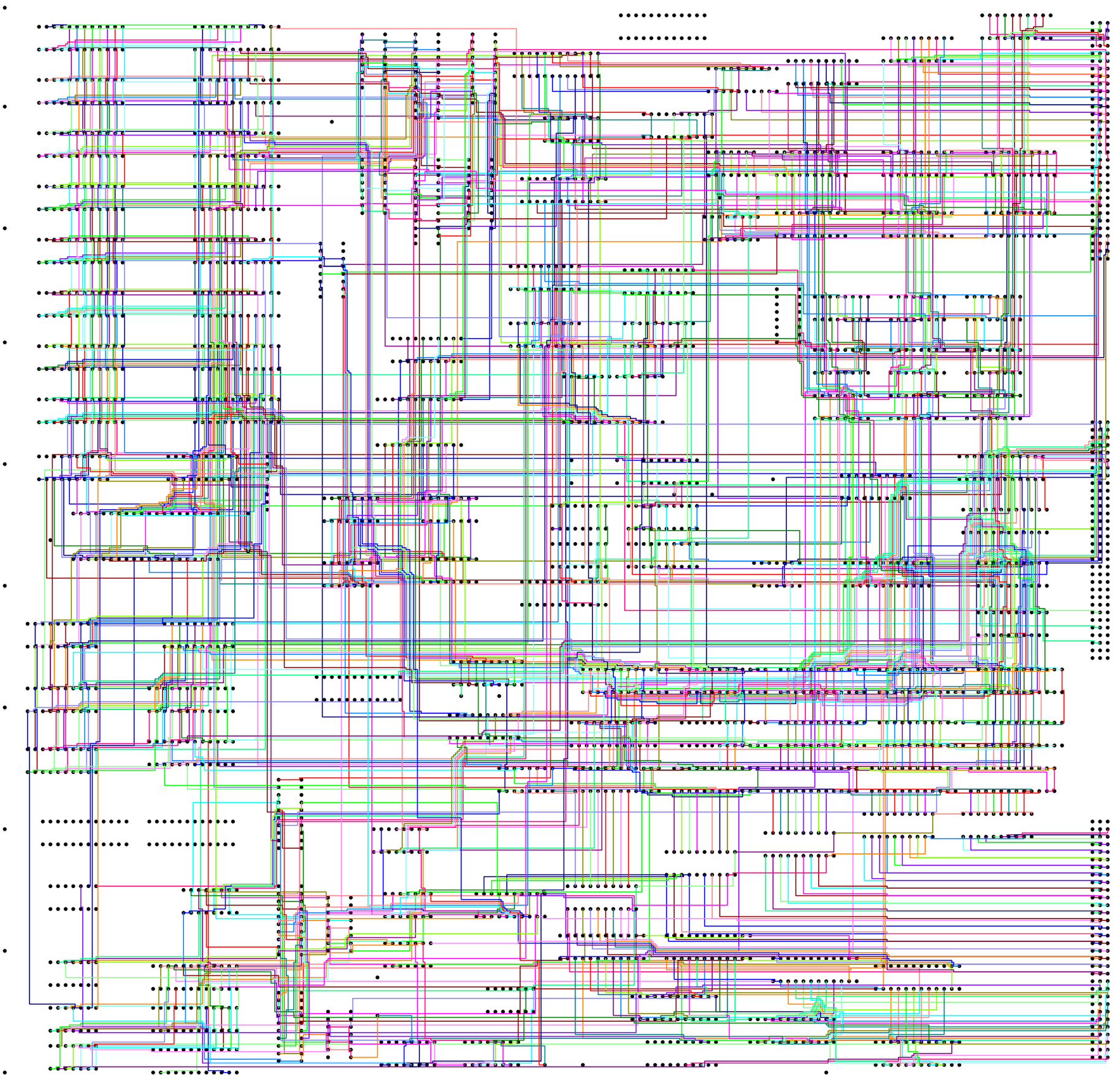
Some kind of shared object - choke point

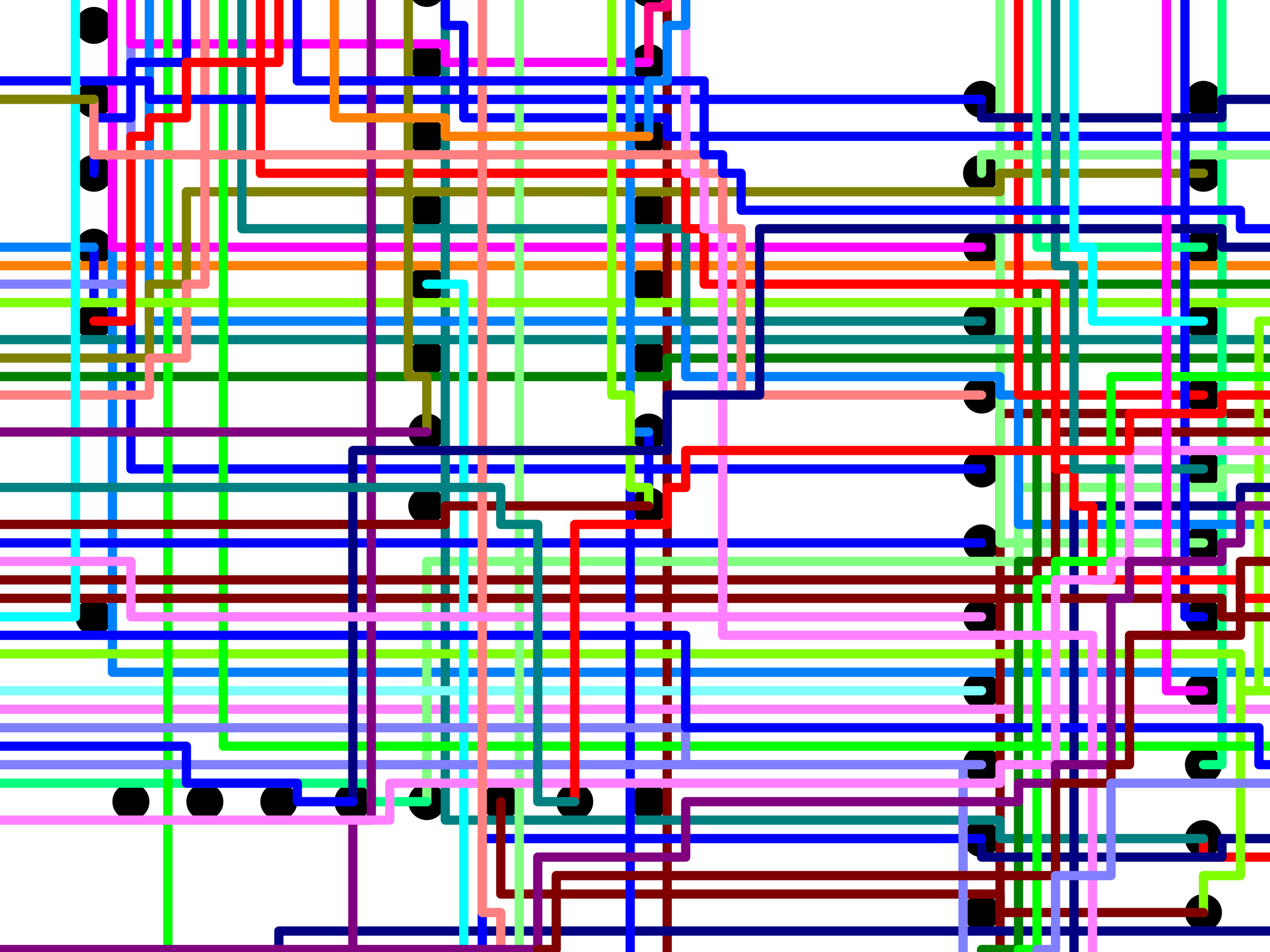


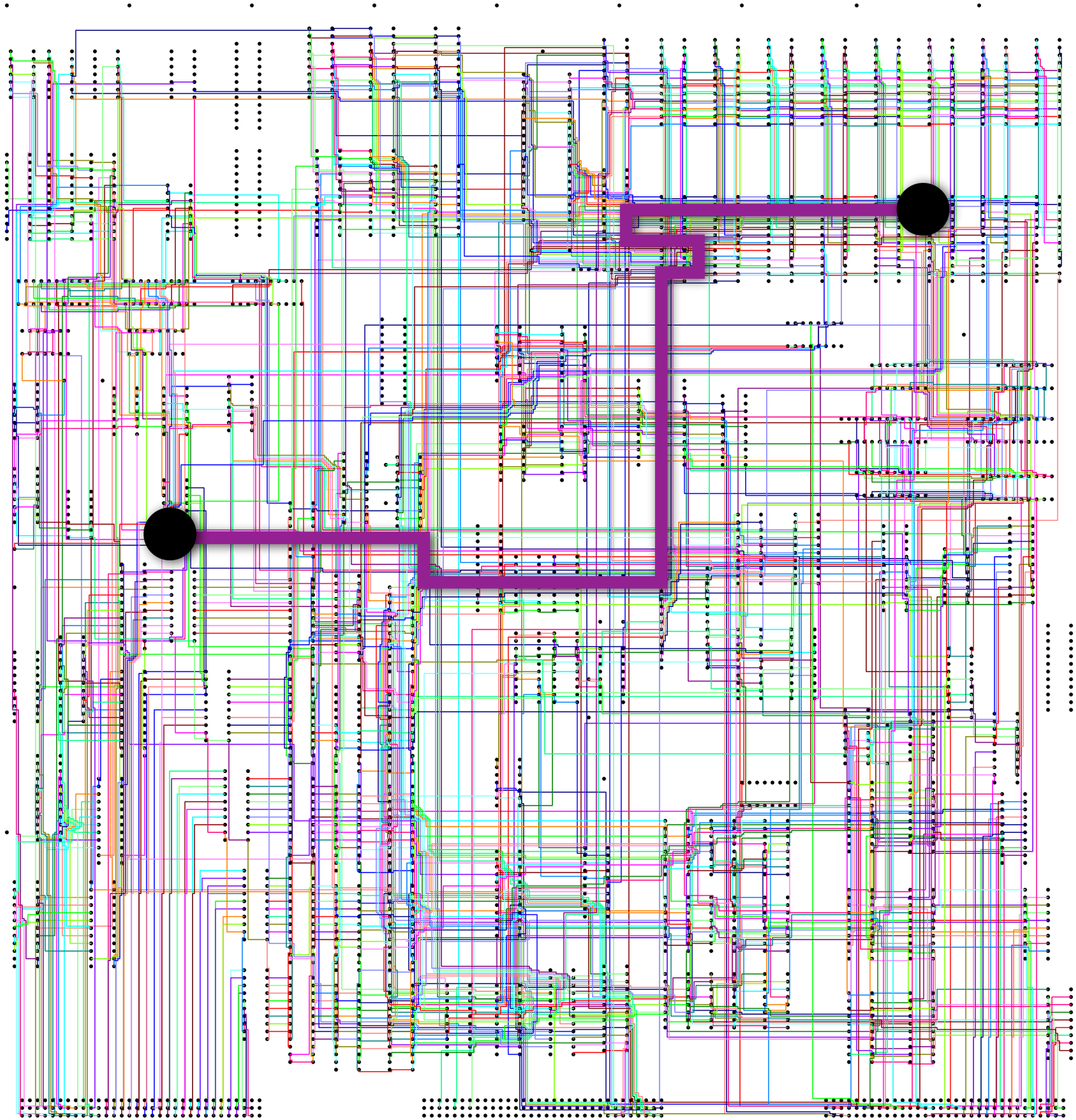
Ideally don't write software with shared objects that cause choke points

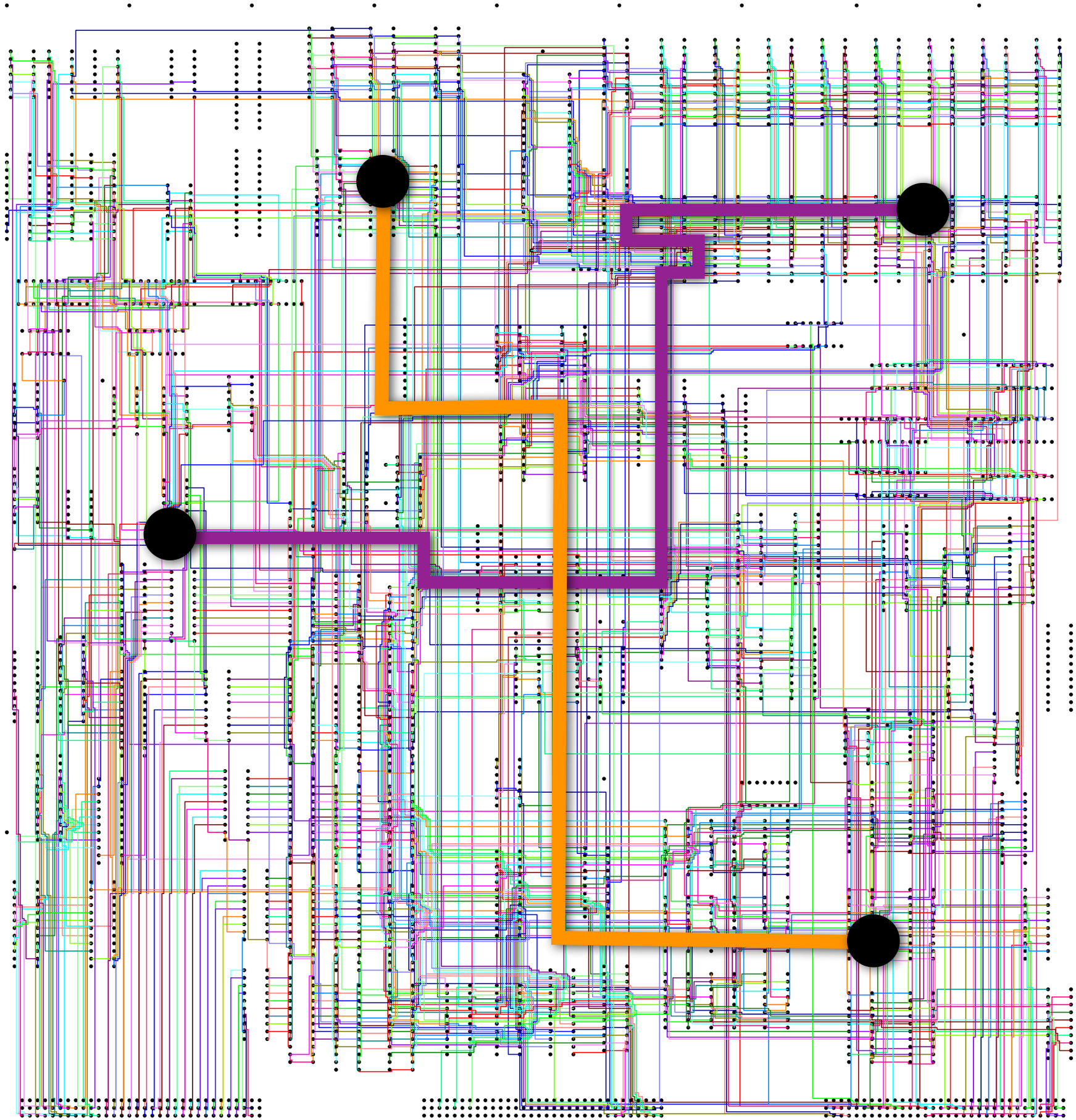
Photo courtesy of Argonne National Laboratory.

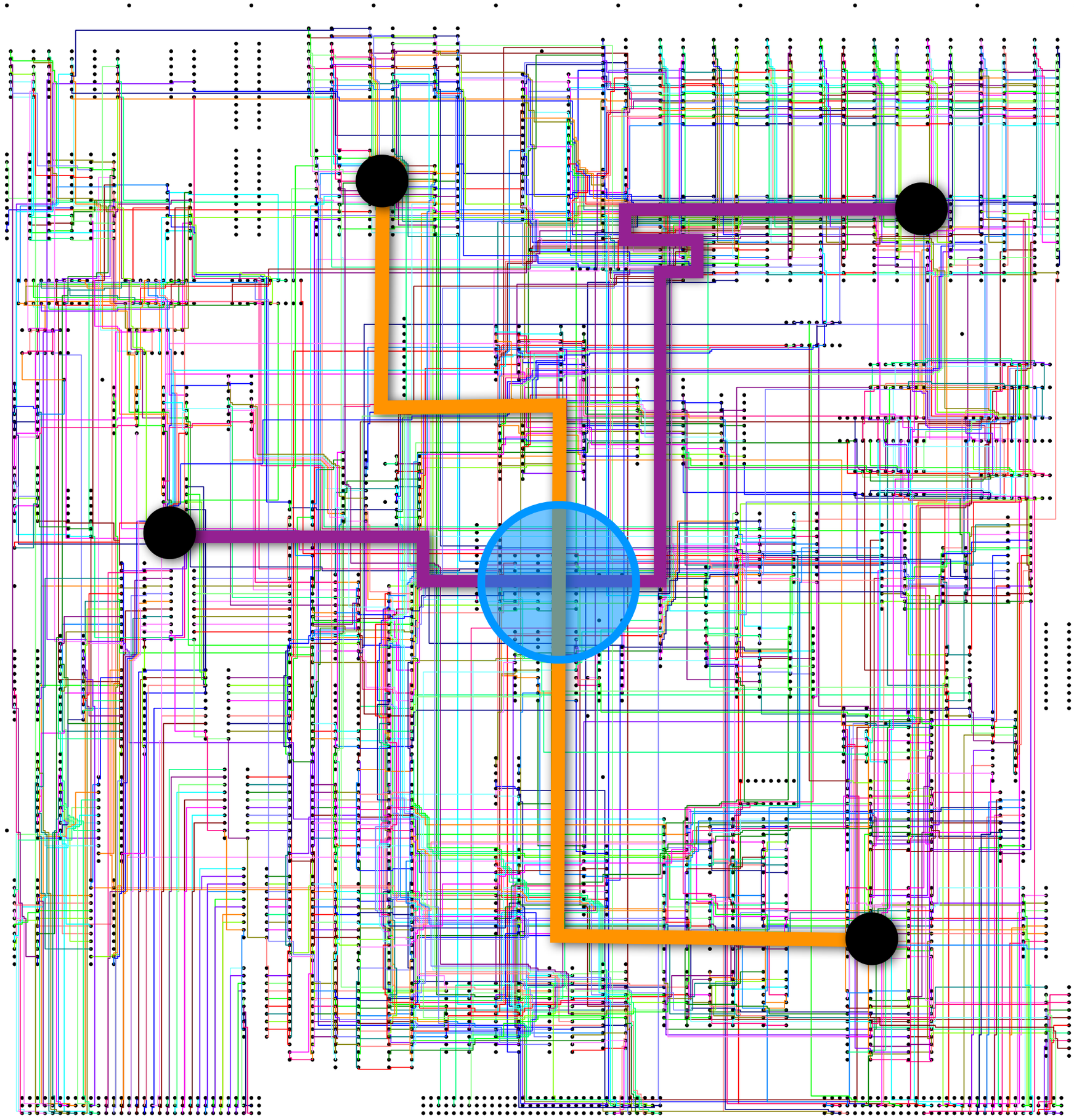
Let's look at a tricky problem

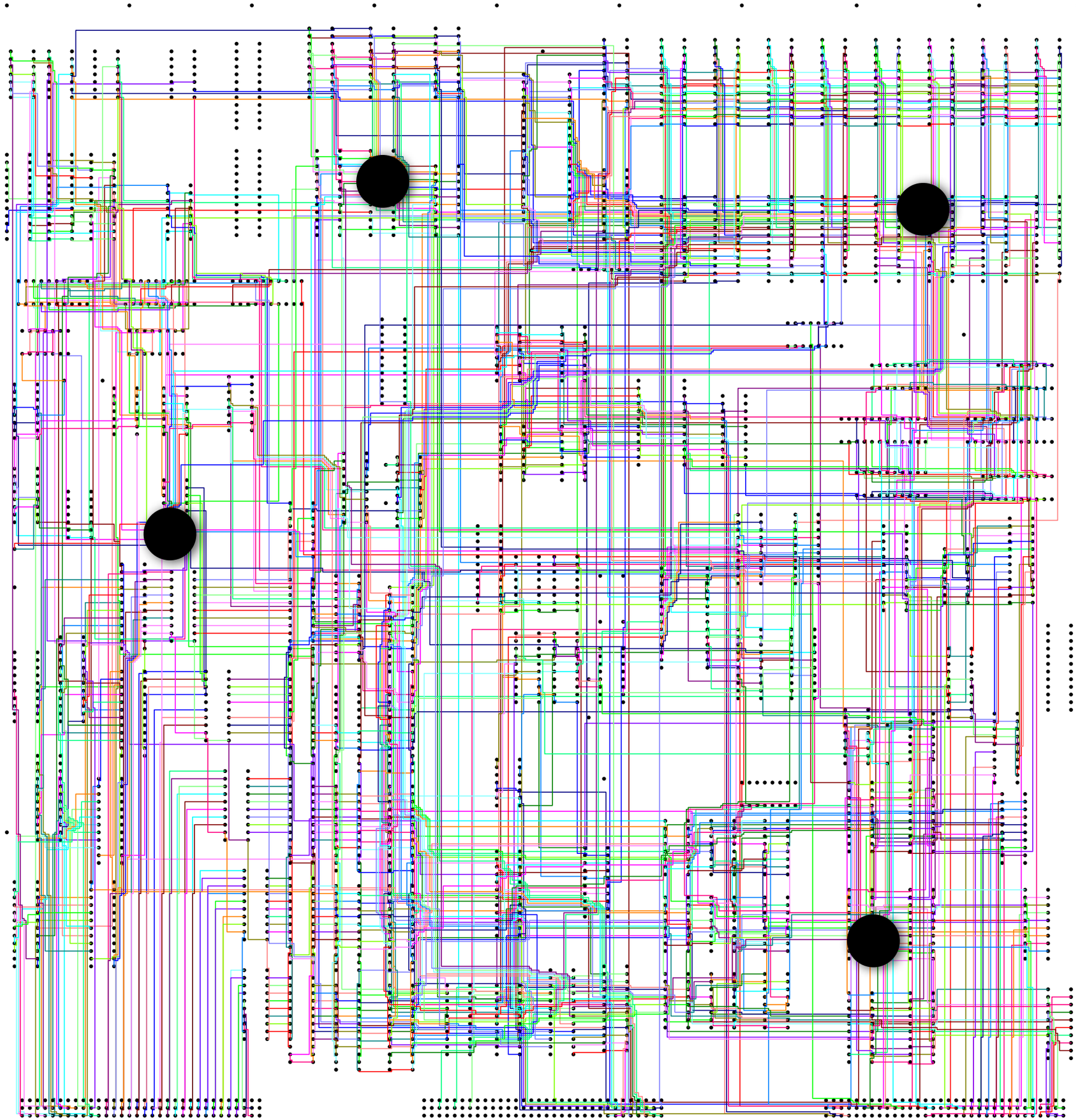


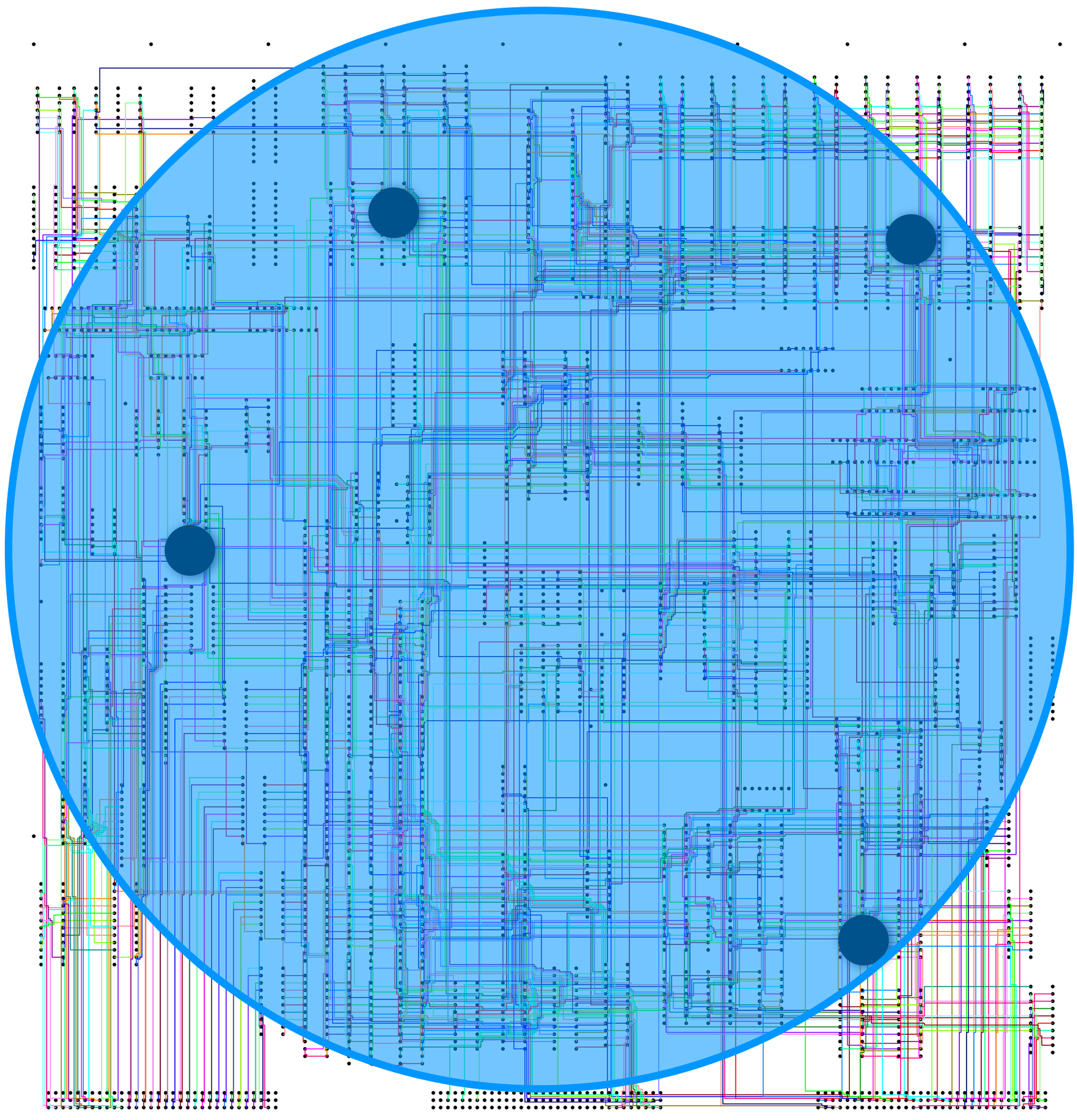




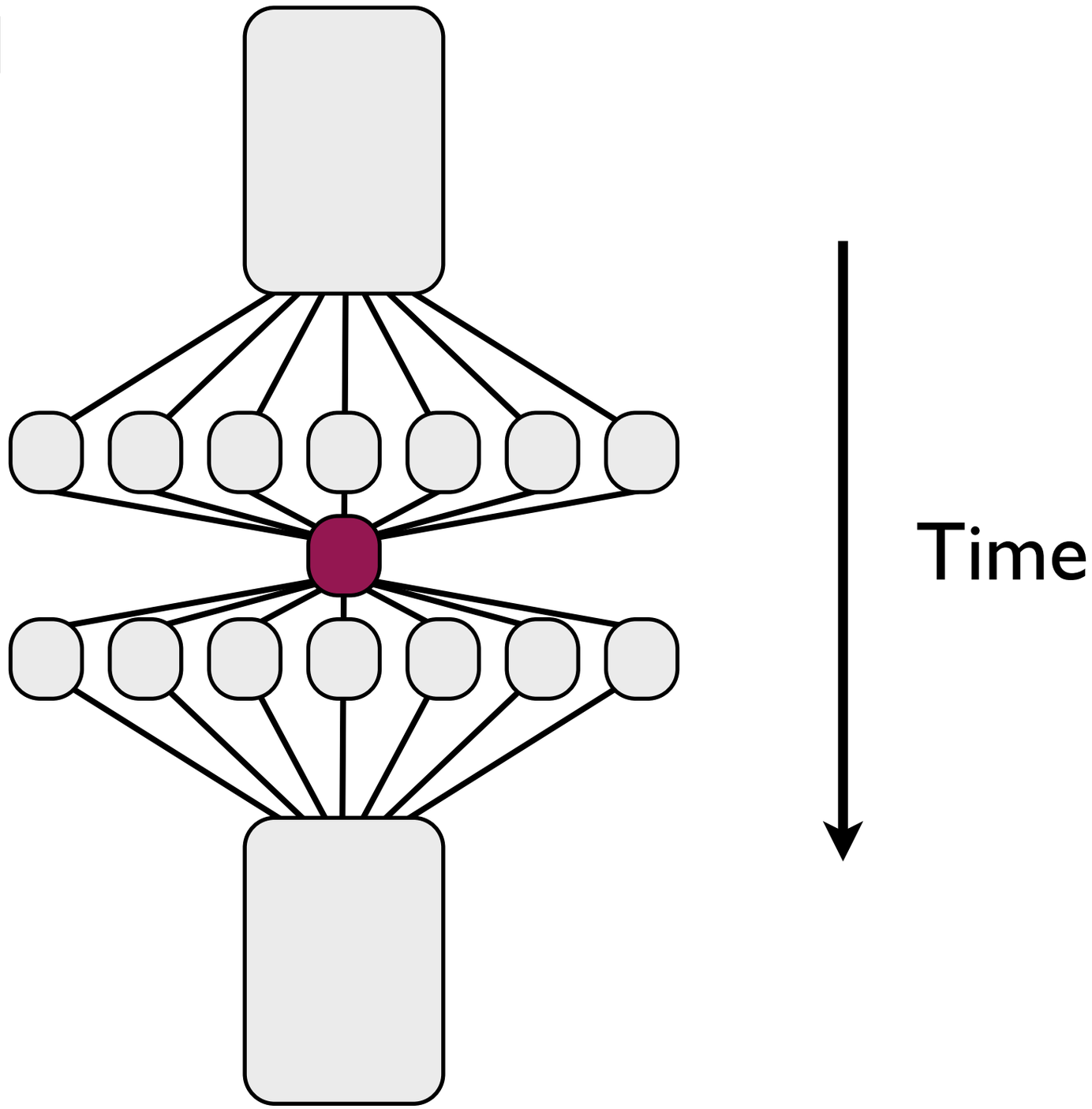








The entire board is one big shared object



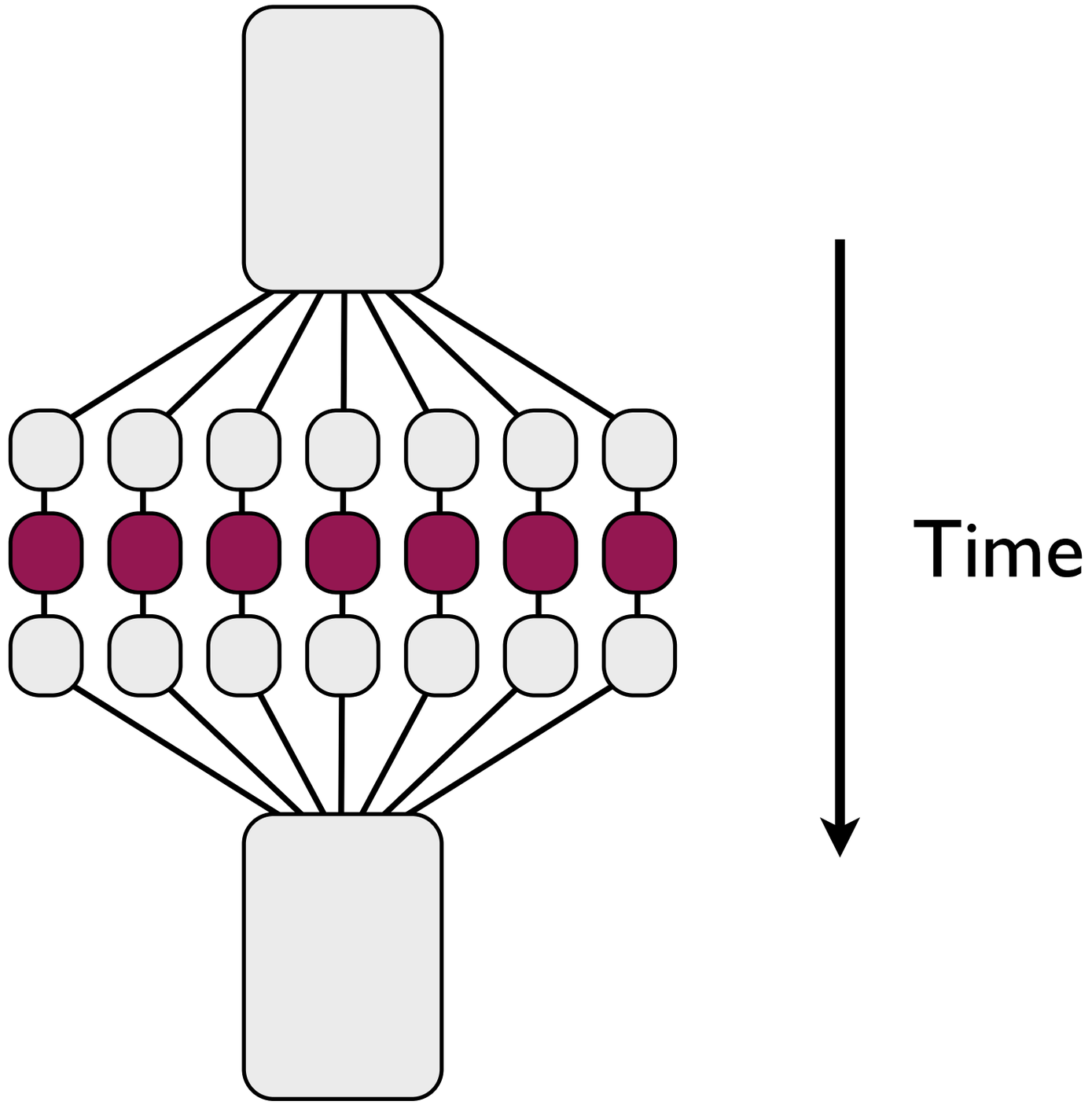
Time

We're calling this an irregular problem - we can't divide up the shared resource before the tasks start

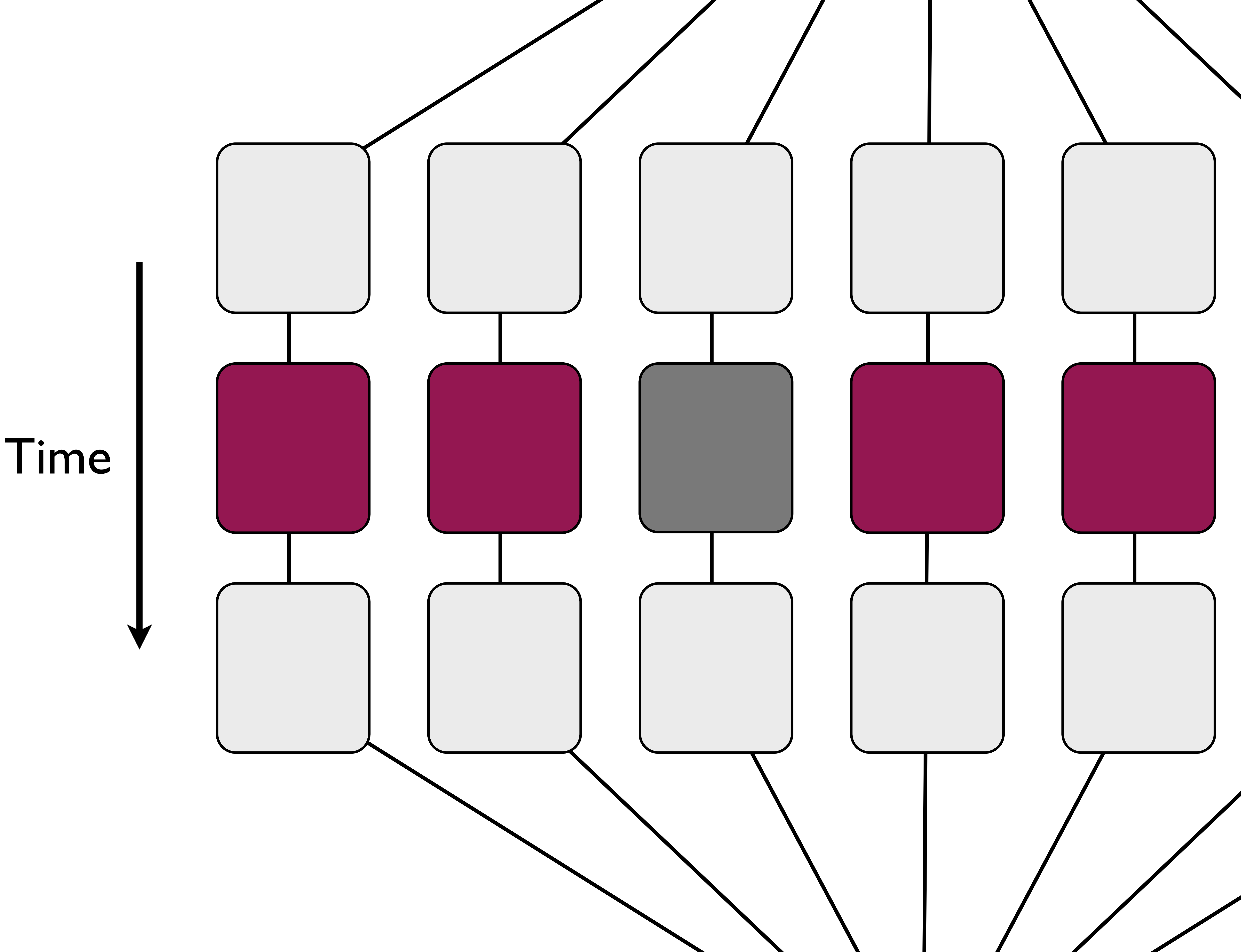
"It's easier to ask forgiveness than it is to get permission."

We'll assume that tasks will not get in each other's way

If they do, we'll sort it out when it happens


Time

Time


Time

A little more time

Two questions:

How can you tell when one task gets in the way of another?

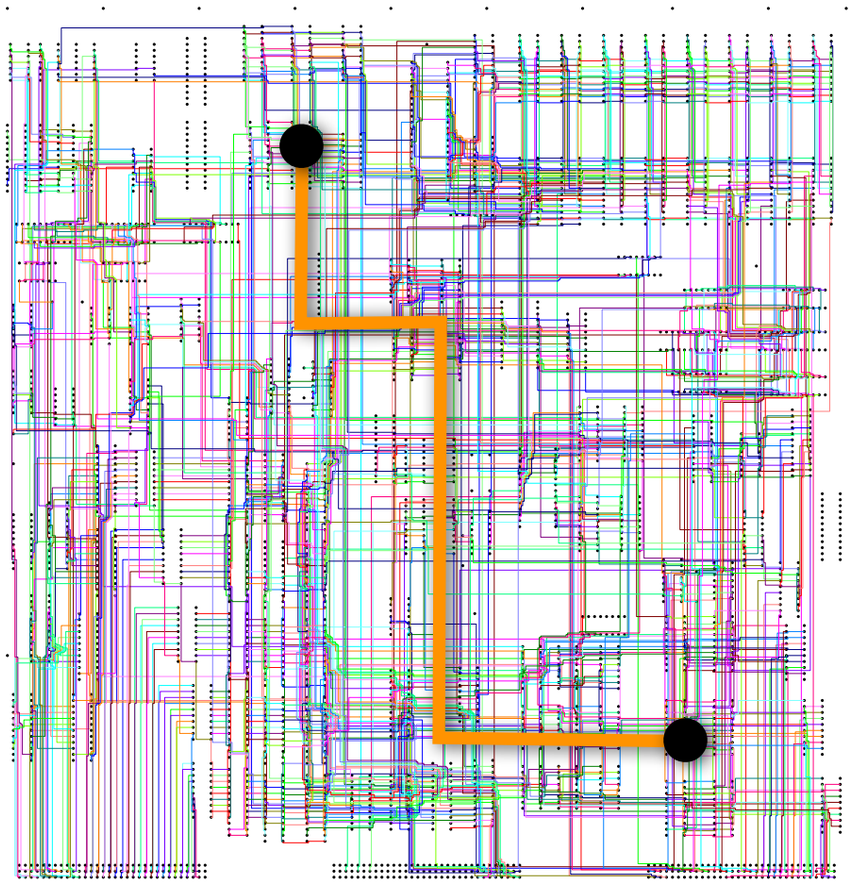
How can you cancel a task that has already been running?

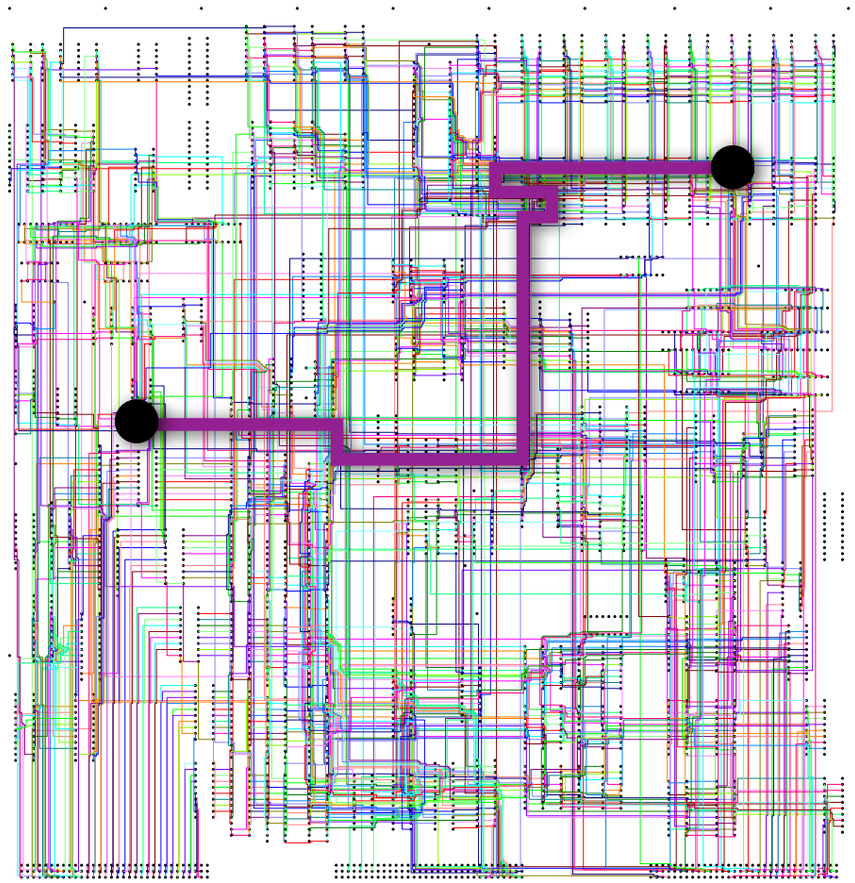
Transactional memory

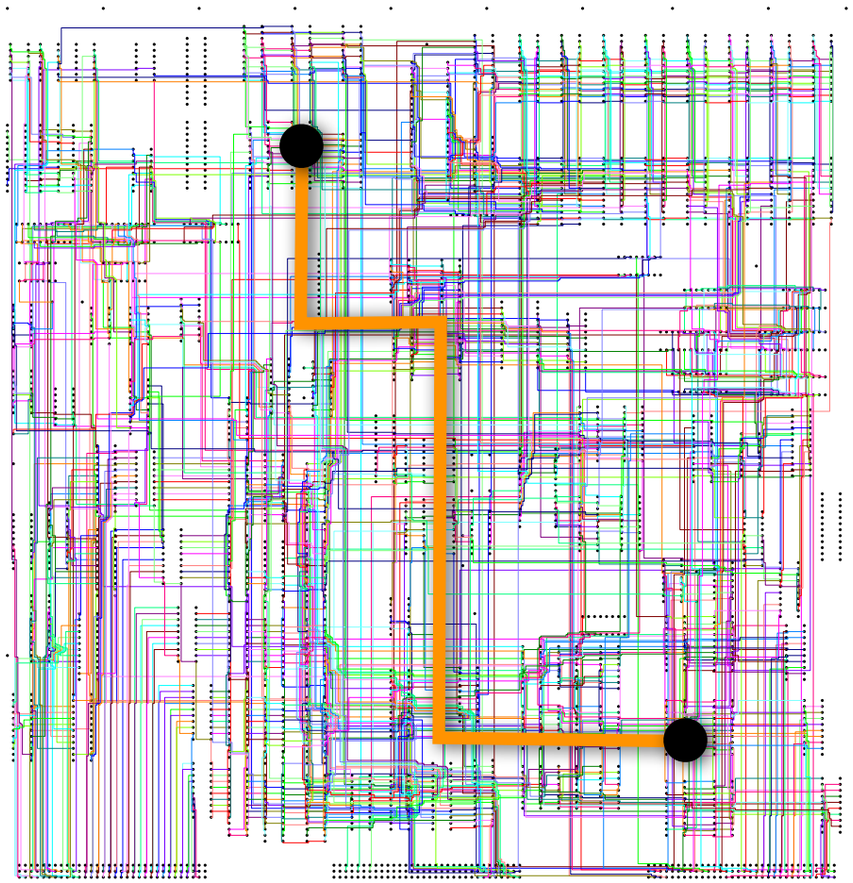
Instead of writing to memory, write to a log

You can tell if two tasks are getting in each other's way by comparing their logs

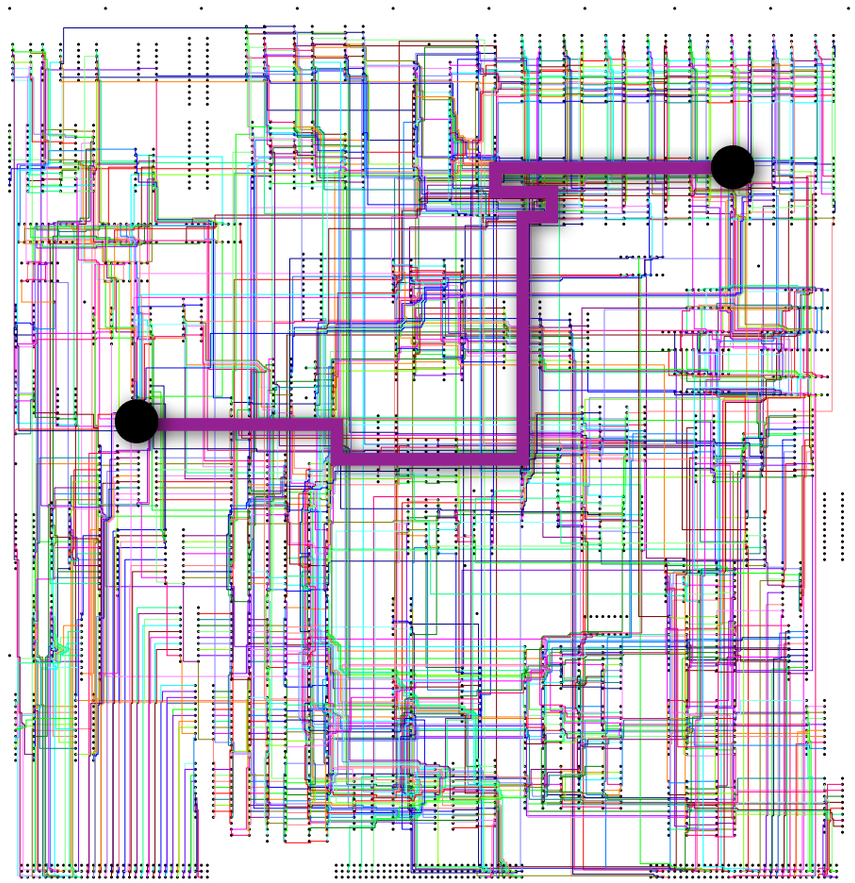
You can cancel a task by throwing the log away



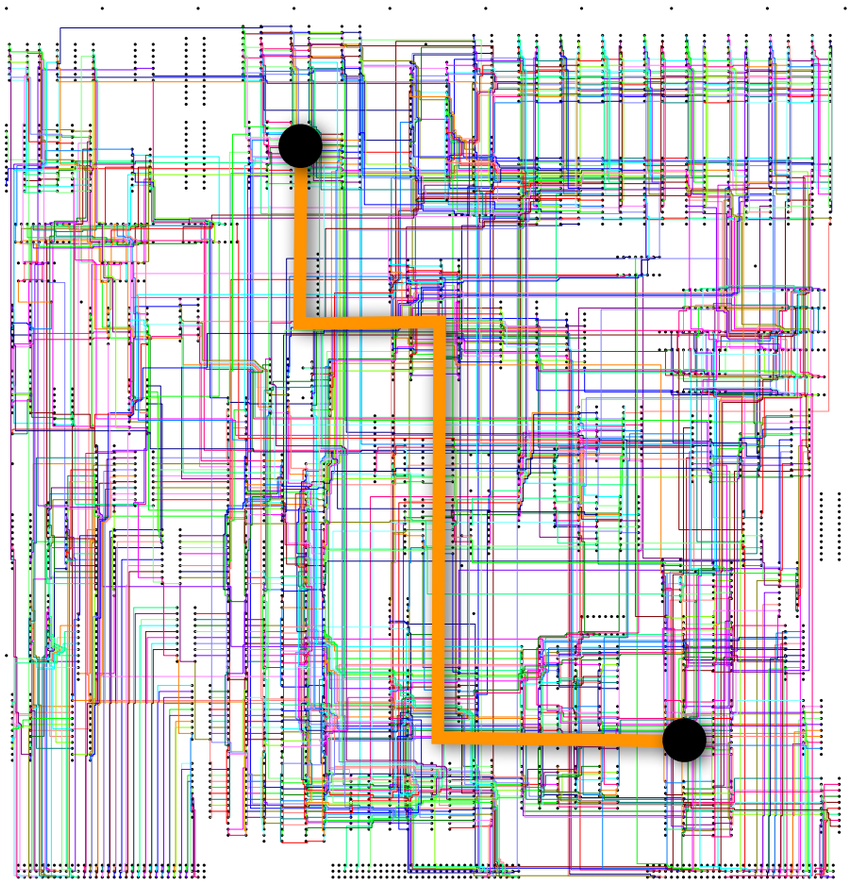


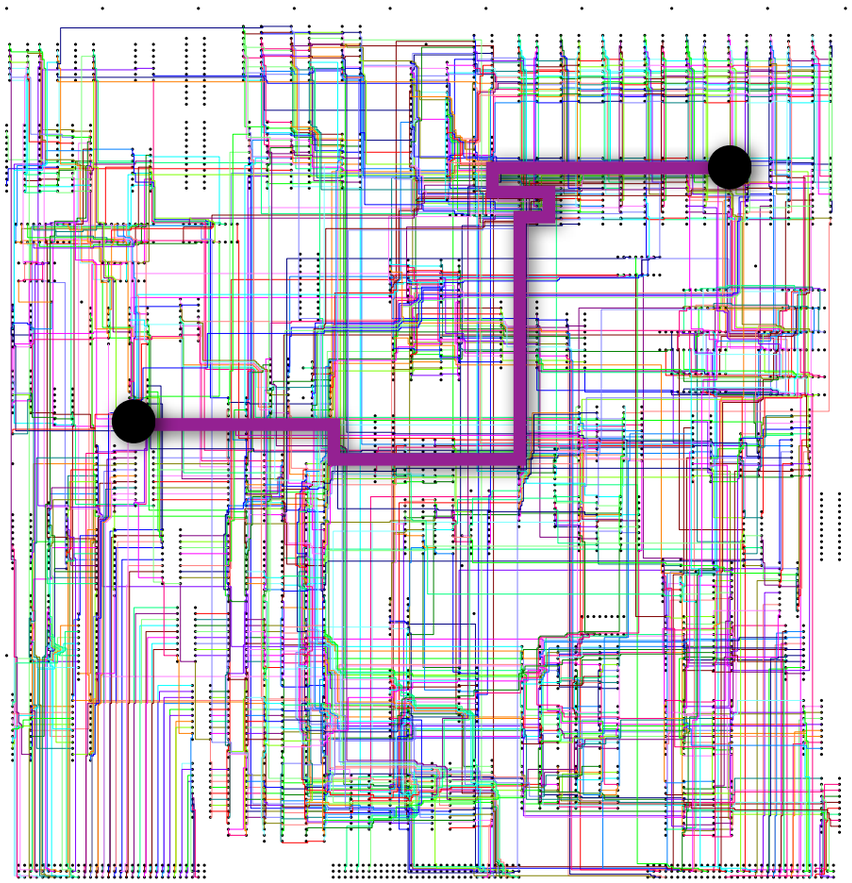


```
0x80d2ef52
0x4ee47f35
0xd6b4eba9
0x2c86d524
0xe617f31d
0x40578fff
0x9bb7febc
0x4ddc0e5f
0xbd660807
```



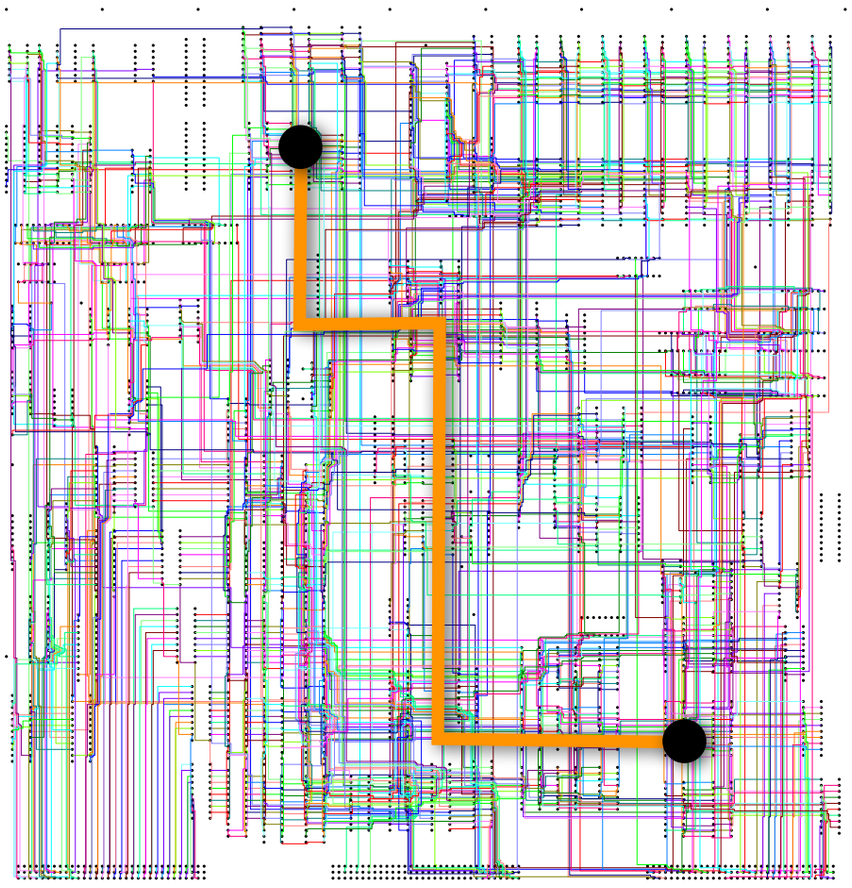
```
0xecb10c62
0xbae0a866
0xe4c2615d
0x61639ad5
0xe617f31d
0x6a8bdcf3
0xe9e88989
0x8fbcf724
0x095b76c0
```



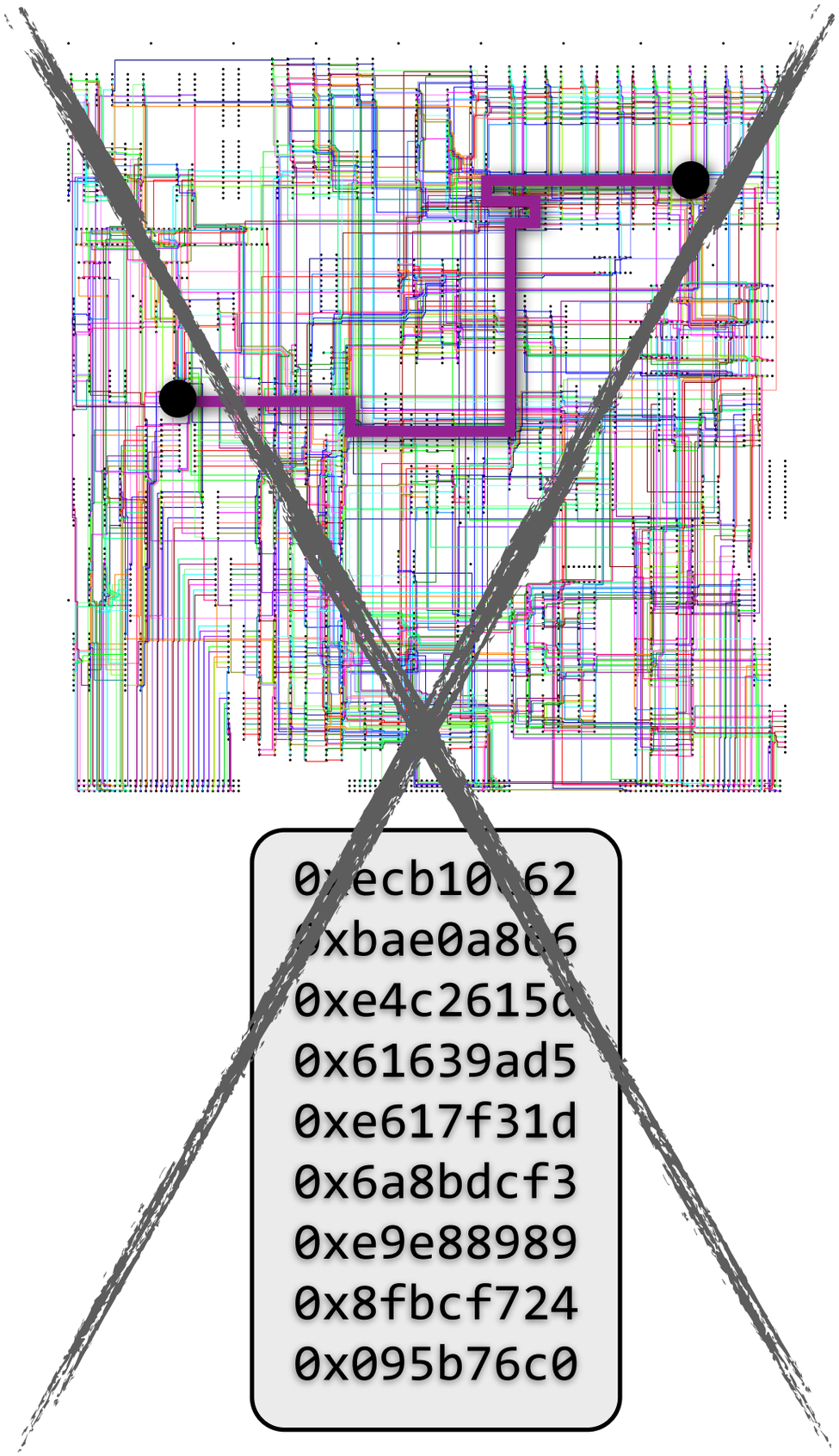


```
0x80d2ef52
0x4ee47f35
0xd6b4eba9
0x2c86d524
0xe617f31d
0x40578fff
0x9bb7febc
0x4ddc0e5f
0xbd660807
```

```
0xecb10c62
0xbae0a866
0xe4c2615d
0x61639ad5
0xe617f31d
0x6a8bdcf3
0xe9e88989
0x8fbcf724
0x095b76c0
```

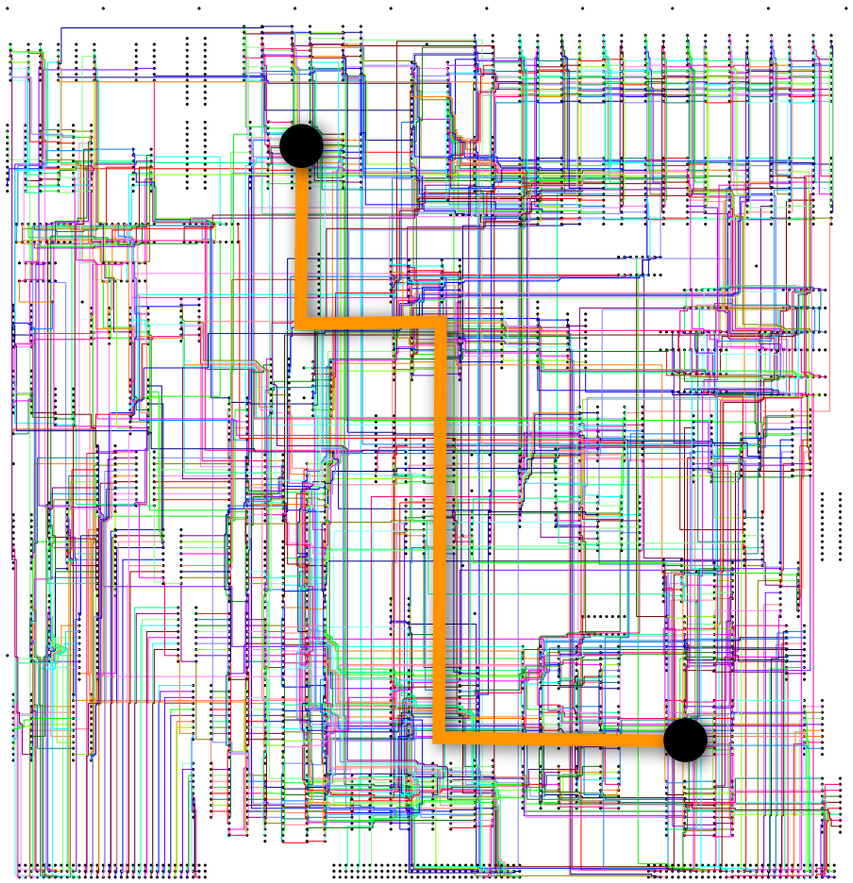


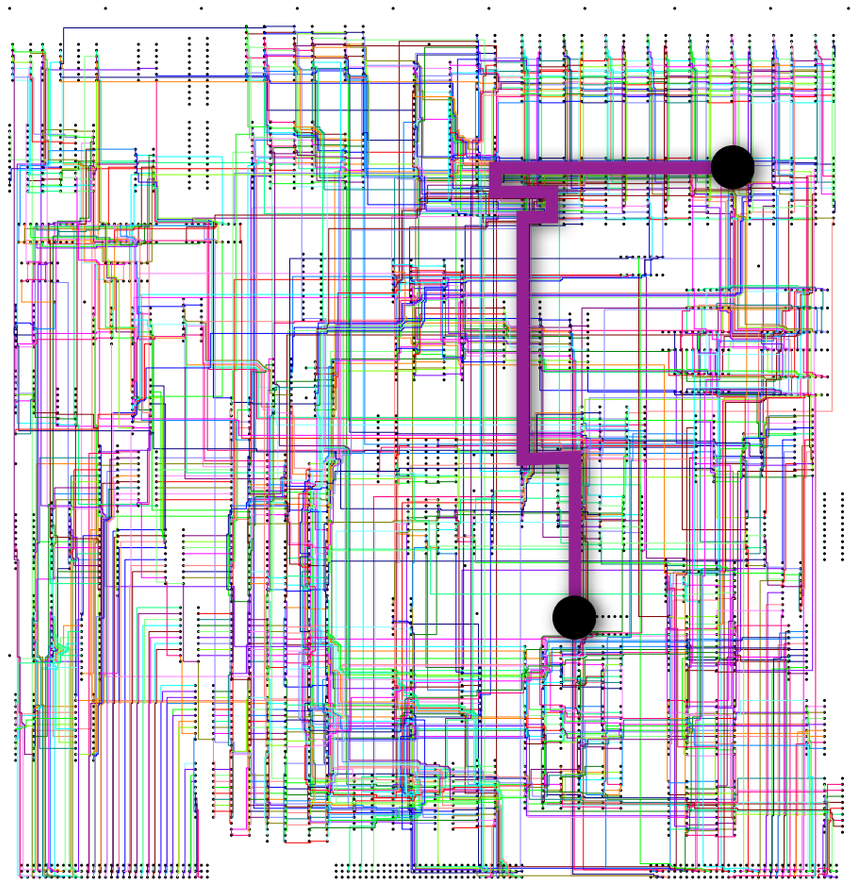
```
0x80d2ef52
0x4ee47f35
0xd6b4eba9
0x2c86d524
0xe617f31d
0x40578fff
0x9bb7febc
0x4ddc0e5f
0xbd660807
```

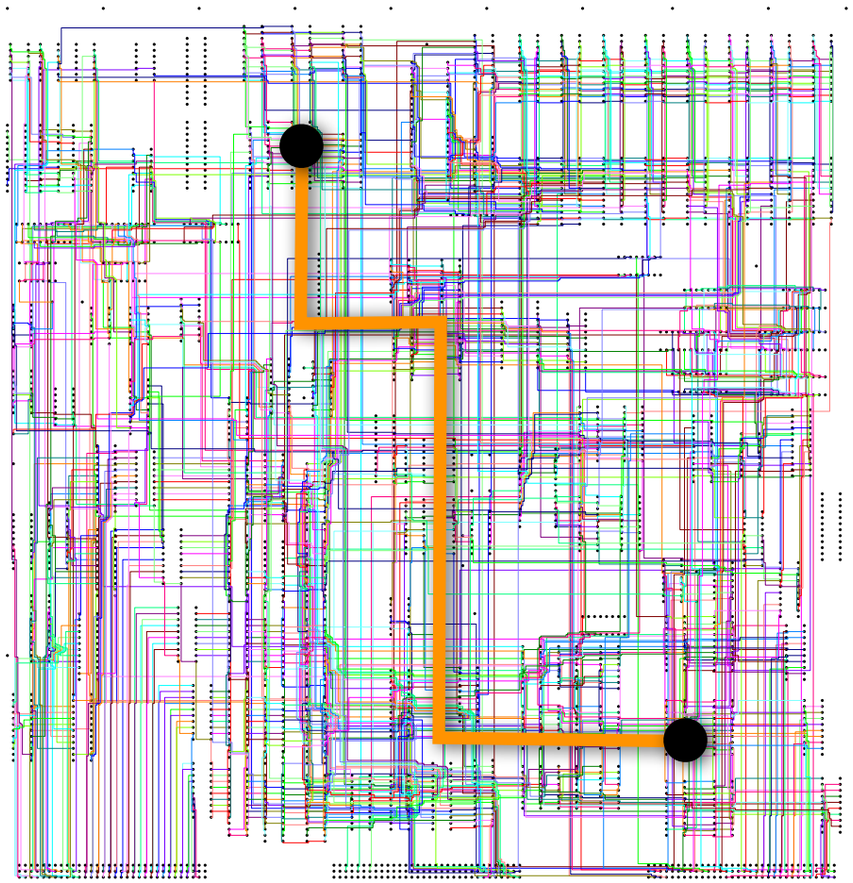


```
0xecb10c62
0xbae0a8c6
0xe4c2615d
0x61639ad5
0xe617f31d
0x6a8bdcf3
0xe9e88989
0x8fbcf724
0x095b76c0
```

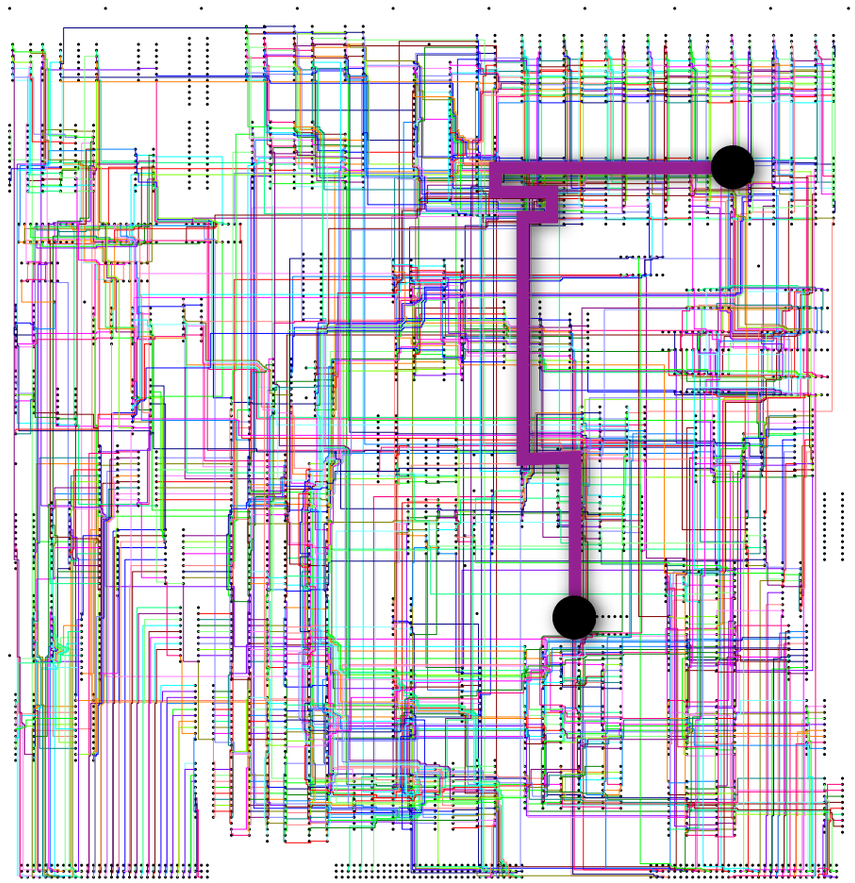
Actually write to memory



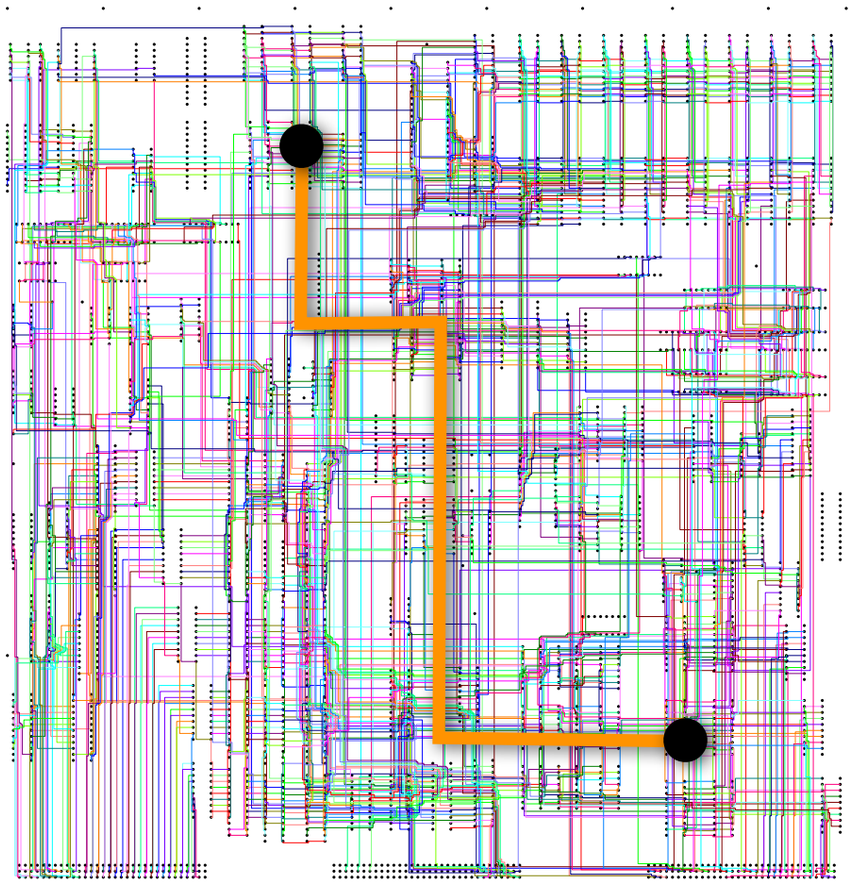




```
0x80d2ef52
0x4ee47f35
0xd6b4eba9
0x2c86d524
0xe617f31d
0x40578fff
0x9bb7febc
0x4ddc0e5f
0xbd660807
```

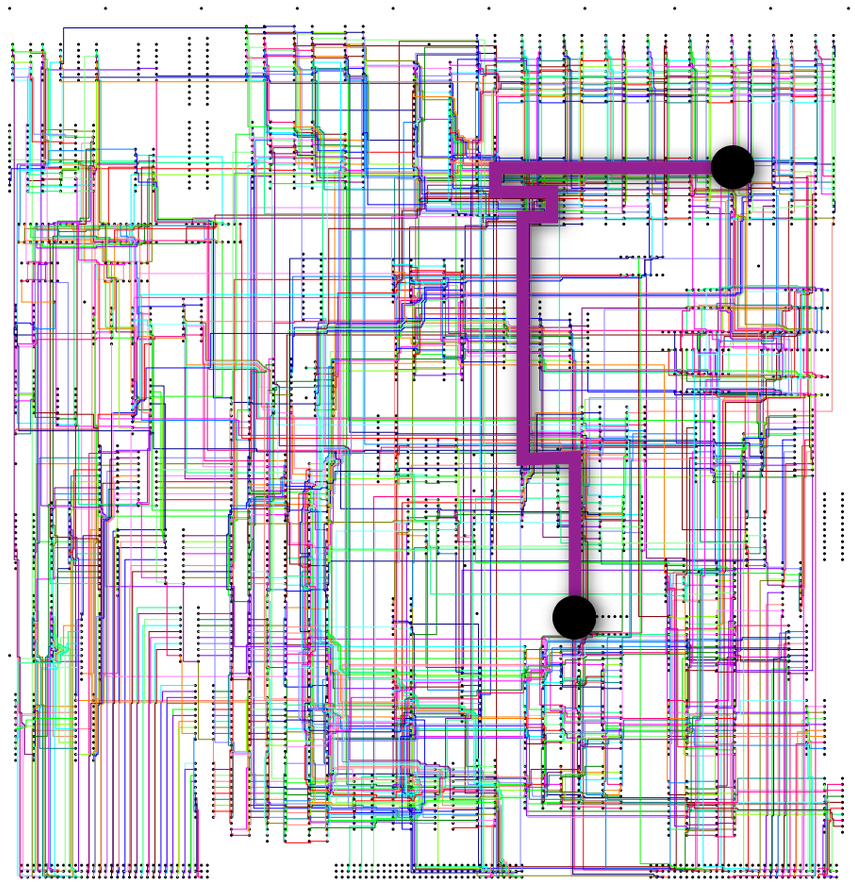


```
0x124683b3
0x60f005b2
0x831327fa
0xf69d8cf9
0x9ea7c8df
0x61f43a4a
0x170c4b44
0x7778a6aa
0x73068a29
```



```
0x80d2ef52
0x4ee47f35
0xd6b4eba9
0x2c86d524
0xe617f31d
0x40578fff
0x9bb7febc
0x4ddc0e5f
0xbd660807
```

Actually write to memory



```
0x124683b3
0x60f005b2
0x831327fa
0xf69d8cf9
0x9ea7c8df
0x61f43a4a
0x170c4b44
0x7778a6aa
0x73068a29
```

Actually write to memory

Transactional memory is moving from research to production

C, C++, Java, Scala, Clojure, Haskell

There are other techniques for reversing computation

If you know you added x to a value, subtract x. If you know you added a node to a graph, remove it.

The Jefferson Time Warp System from the mid-80s - send anti-messages over networks.

Do we have a solution?

Transactional memory can be slow

The hardware is probably limited

Not the magic bullet some hoped

Optimistic execution in general can be wasteful

Irregular problems are the billion dollar problems

facebook

Google™

[1]

Physical meshes
Web and social graphs
Machine learning networks
Data mining

[1] J. Tournois, C. Wormser, P. Alliez, and M. Desbrun. *Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation*. Technical Report 6826, INRIA, 2009.

Chris Seaton
seatonc@cs.man.ac.uk
chrisseaton.com