Hardware
- CPUs
- GPUs / accelerators
- Network
- Disk
- Memory RAM / caches

Software
- algorithms
- PL / runtime sys.
- OS
- architectures
  - MT
  - MP
  - event-based

Making Sense of Perf in Data Analytics Frameworks ☺

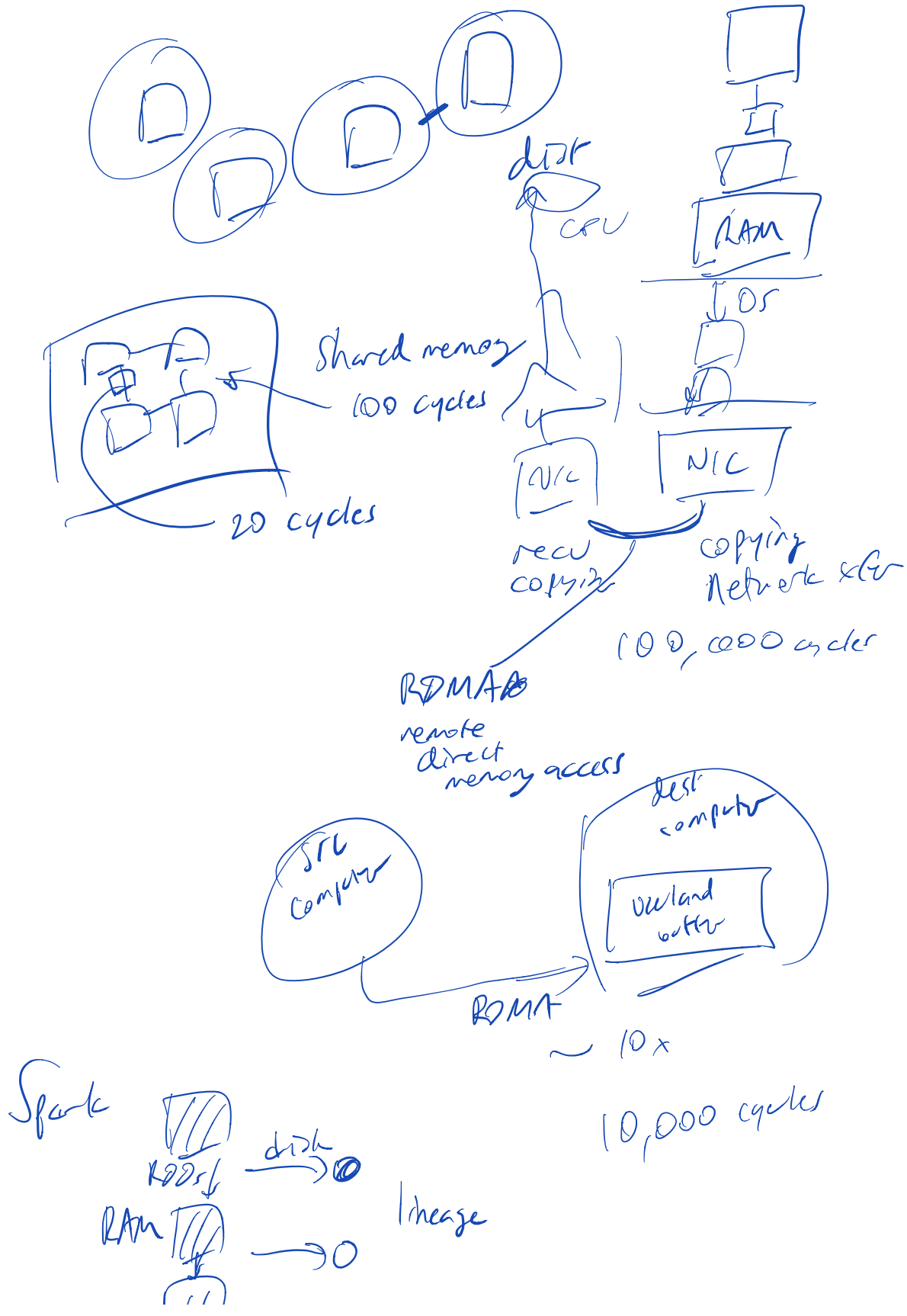
blocked time analysis

infinite resources

Spark SQL

CPU "bound"

I/O bound
- disk bound
- network bound


disk
CPU
RAM
OS
Shared memory
100 cycles
NIC
NIC
20 cycles
recv copying
copying
Network xfer
100,000 cycles
RDMA
remote
direct
memory access
dest computer
src computer
user land buffer
RDMA
~ 10x
10,000 cycles
Spark
RDDs
disk
RAM
lineage

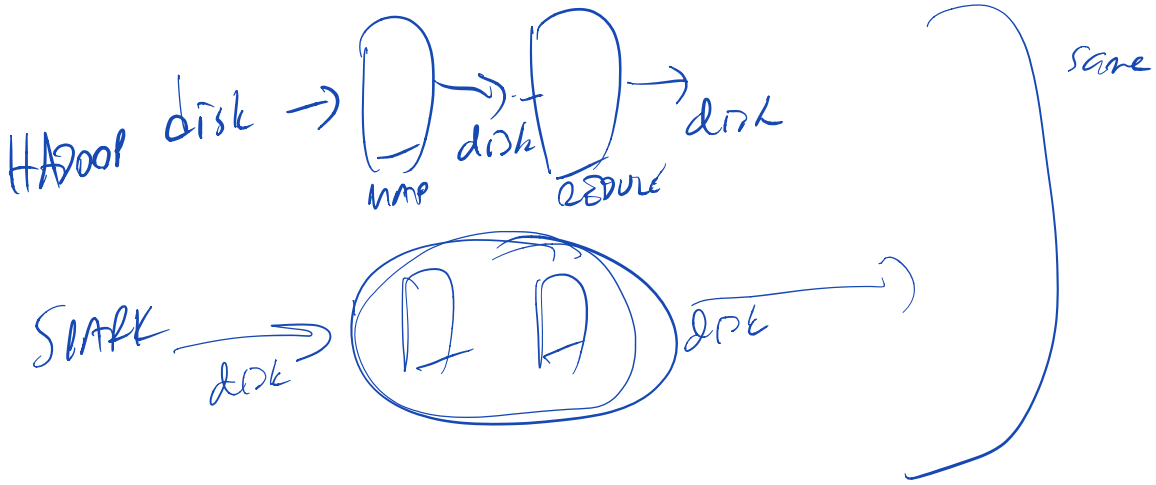
1/41

Spark vs. Hadoop

not always better

1. data too large to fit in RAM
   workload 2TB
   RAM-capacity 100 GB
   "paging"

2. no iteration

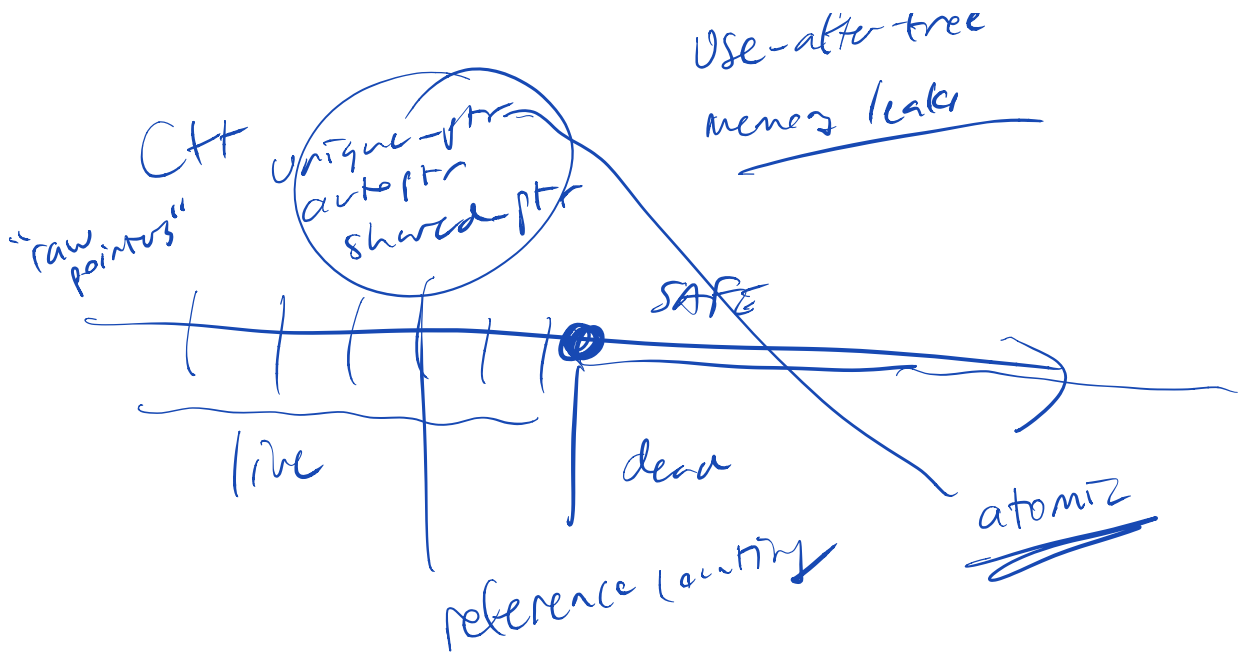


perf. workload dependent

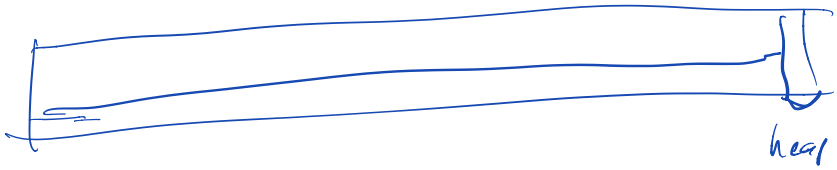
GC remains a problem

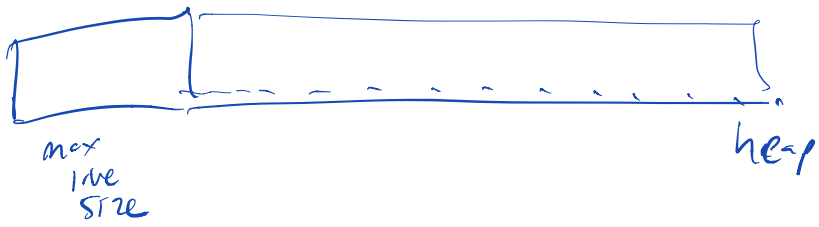
Google → "MapReduce" C++

GC → safety



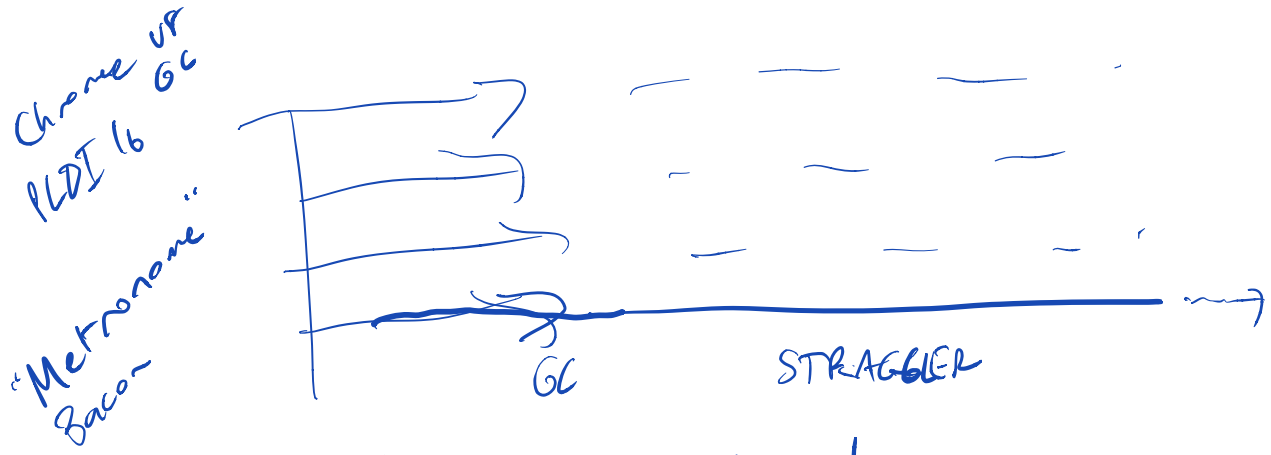
GC pause times
space overhead





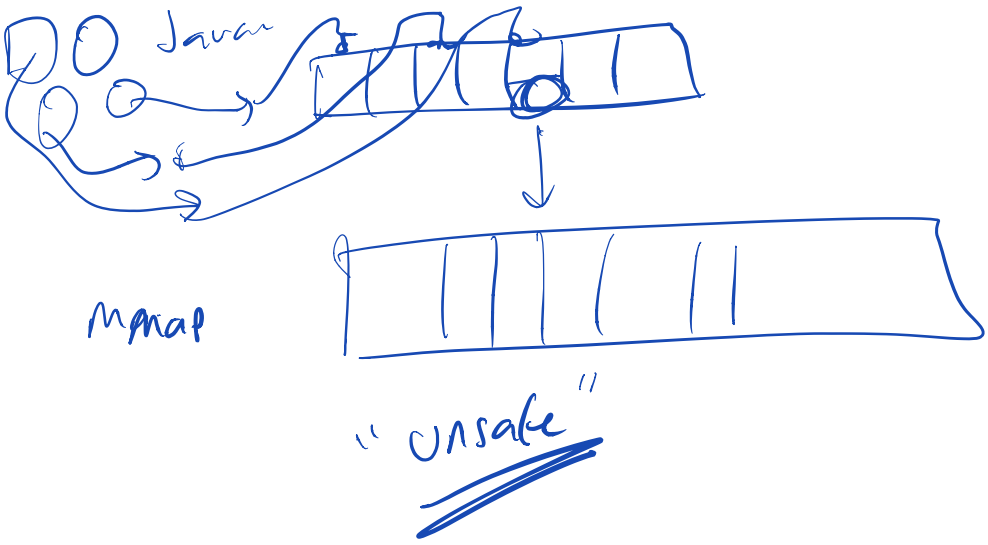
~ 3-5x more than using malloc/free

full heap GC → pause times



Project Tungsten — manual memory mgmt in Spark!

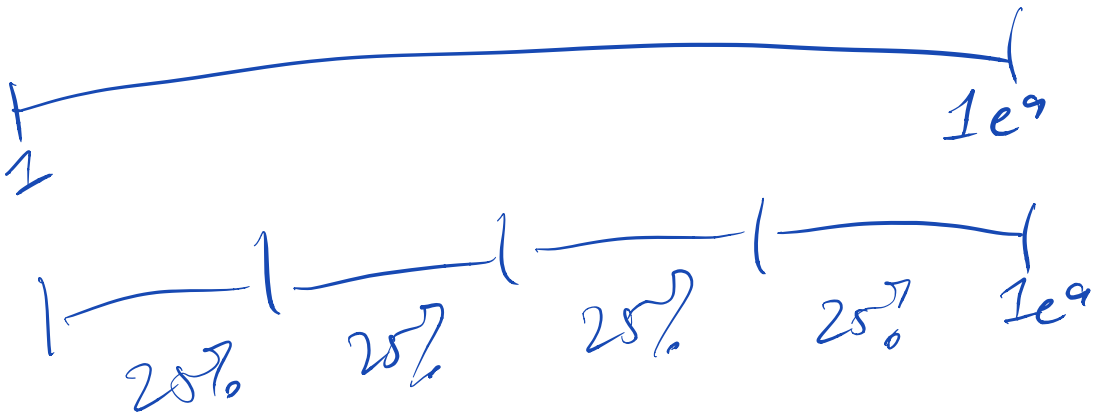
Goal → set n/e of GC

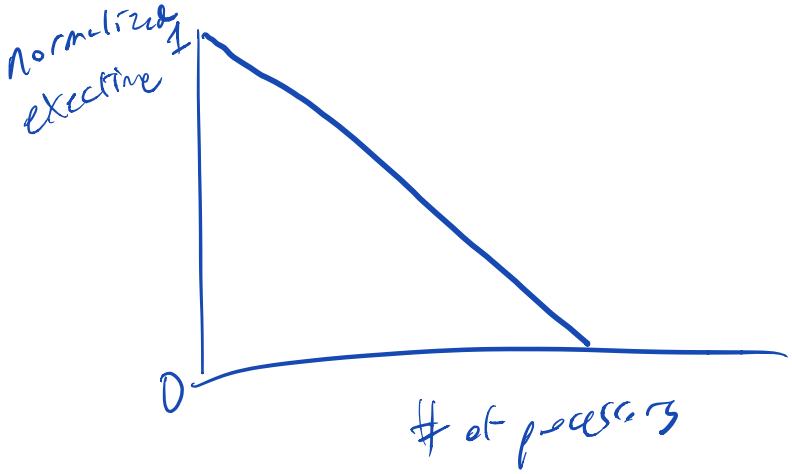


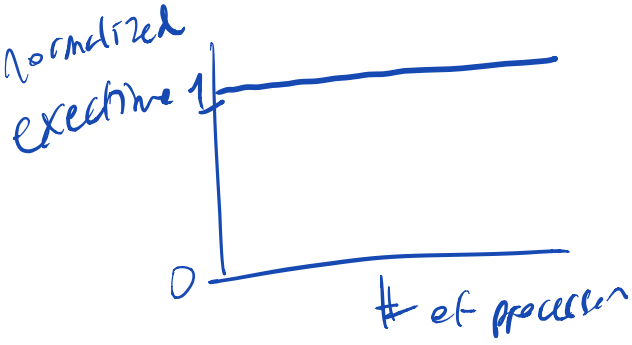
$$\sum_{i=1}^{1e^9} i$$

```
s=0
for(i=1; i<1e9; i++) {
  s+=i;
}
return s;
```

comp







$$S = \frac{1e^9 * (1e^9 + 1)}{2}$$

baseline: fastest state-of-the-art version on one node

COST Config. Outperforms a Single Thread

max(∞, # of cores required to match perf. s.t. version)

---

problems -

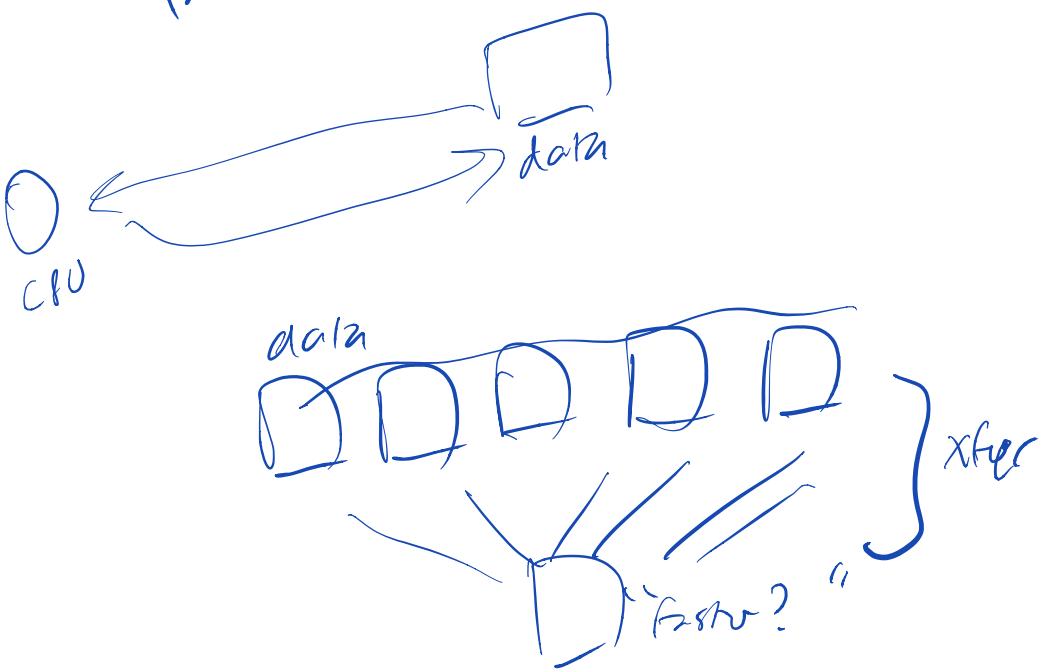
too big to fit in RAM

too big to fit on disk

~Gustafson's Law

algorithms for single thread ≠ algs. for dist comp.

fault tolerance (solvable)



Domain-Specific Languages

SQL

Excel

"analysts"

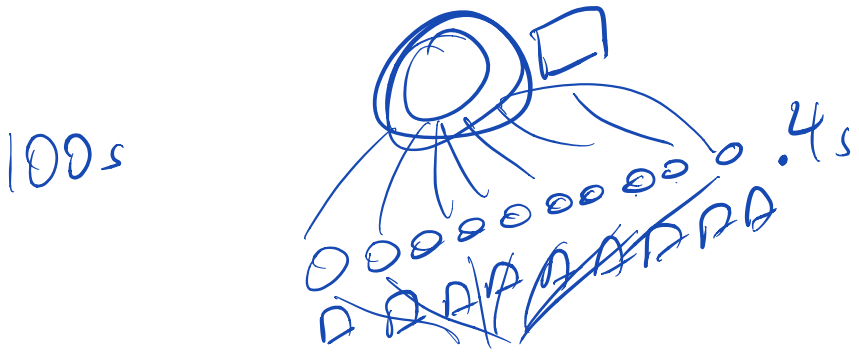
libraries ≈ DSL

Java
JS
Python

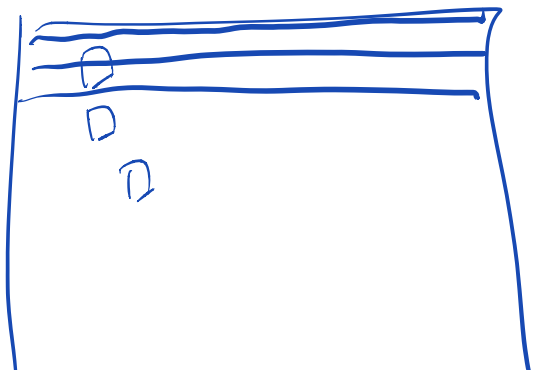
general-purpose

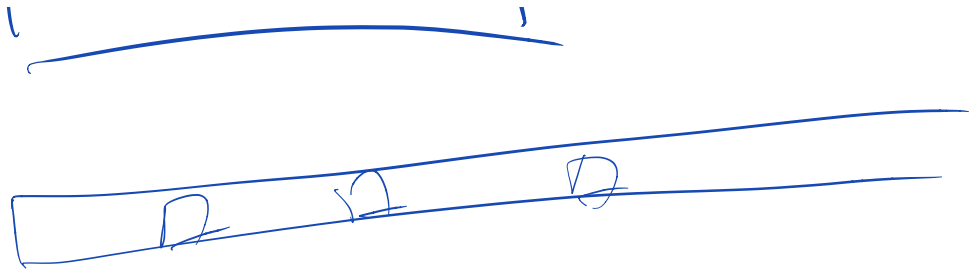
s.t. → s.t.! ONE THREAD

multiple processors



superlinear speedup





Space-filling curve

Z-curve