DASlab

Home | Syllabus | Schedule | Project | Self Evaluation | Class Notes | Forum & Announcements

HARVARD
School of Engineering
and Applied Sciences

# CS165
## Data Systems

Have fun learning to design and build modern data systems

class 13

# fast scans 1.0

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/

Tuesday Research session
**shared scans and scans vs indexes** (project=m2,m3)

Wednesday section
**performance tools** (all project milestones)

Thursday section (extra), 6:30-8pm, Pierce, 100F
**midterm review**

Sunday OH with Stratos, 2-6:30pm
**milestone 2 & 3**

milestone 5 becomes bonus

milestone 2 tests are up
& automated testing + leaderboard start today


hardware, data and query based optimizations

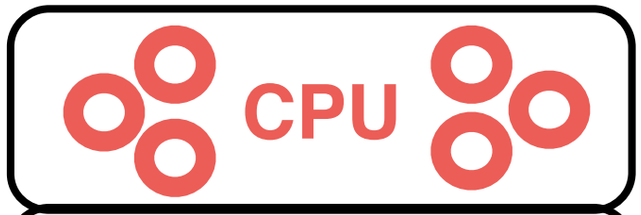
(project=m3)

**apply to all**
**algo/data structures**

CPU

memory

**Vs.**




registers

on chip cache

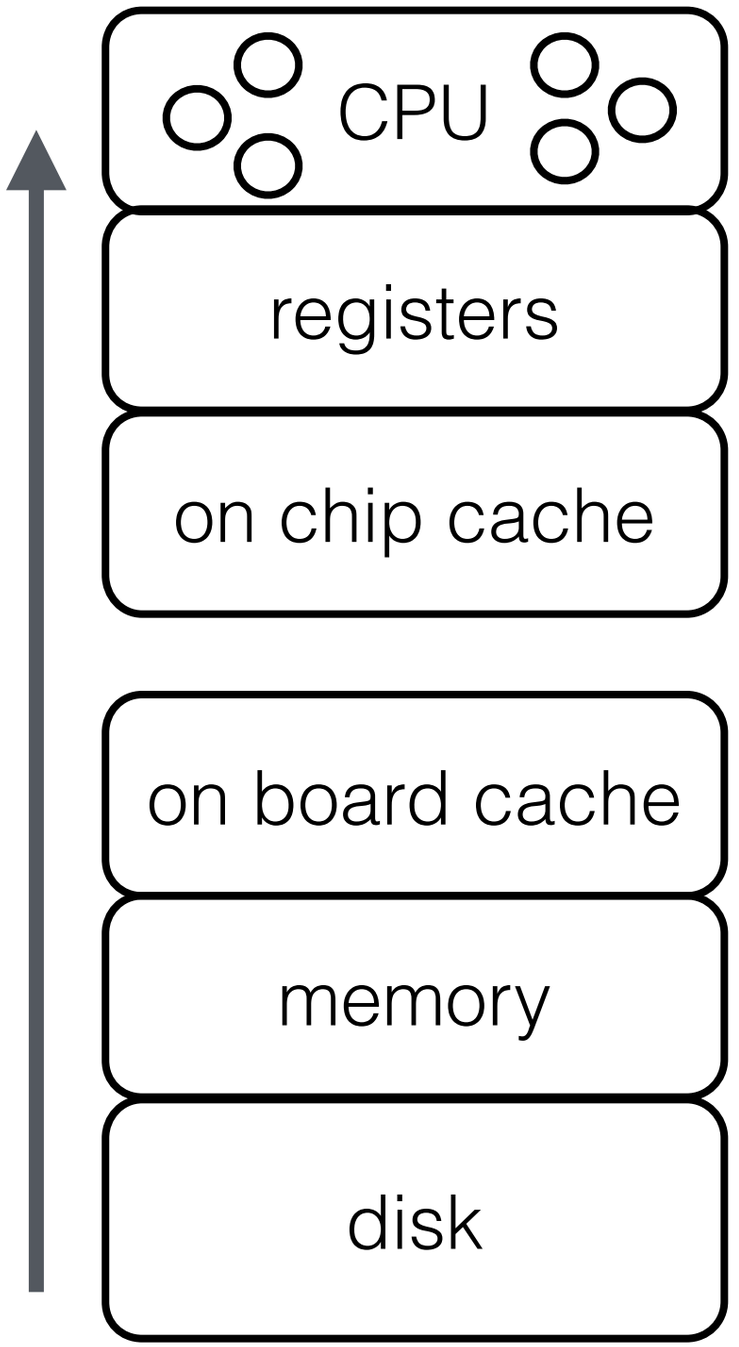
on board cache

memory

disk

always want to minimize
data movement - computation

**& utilize all resources!**

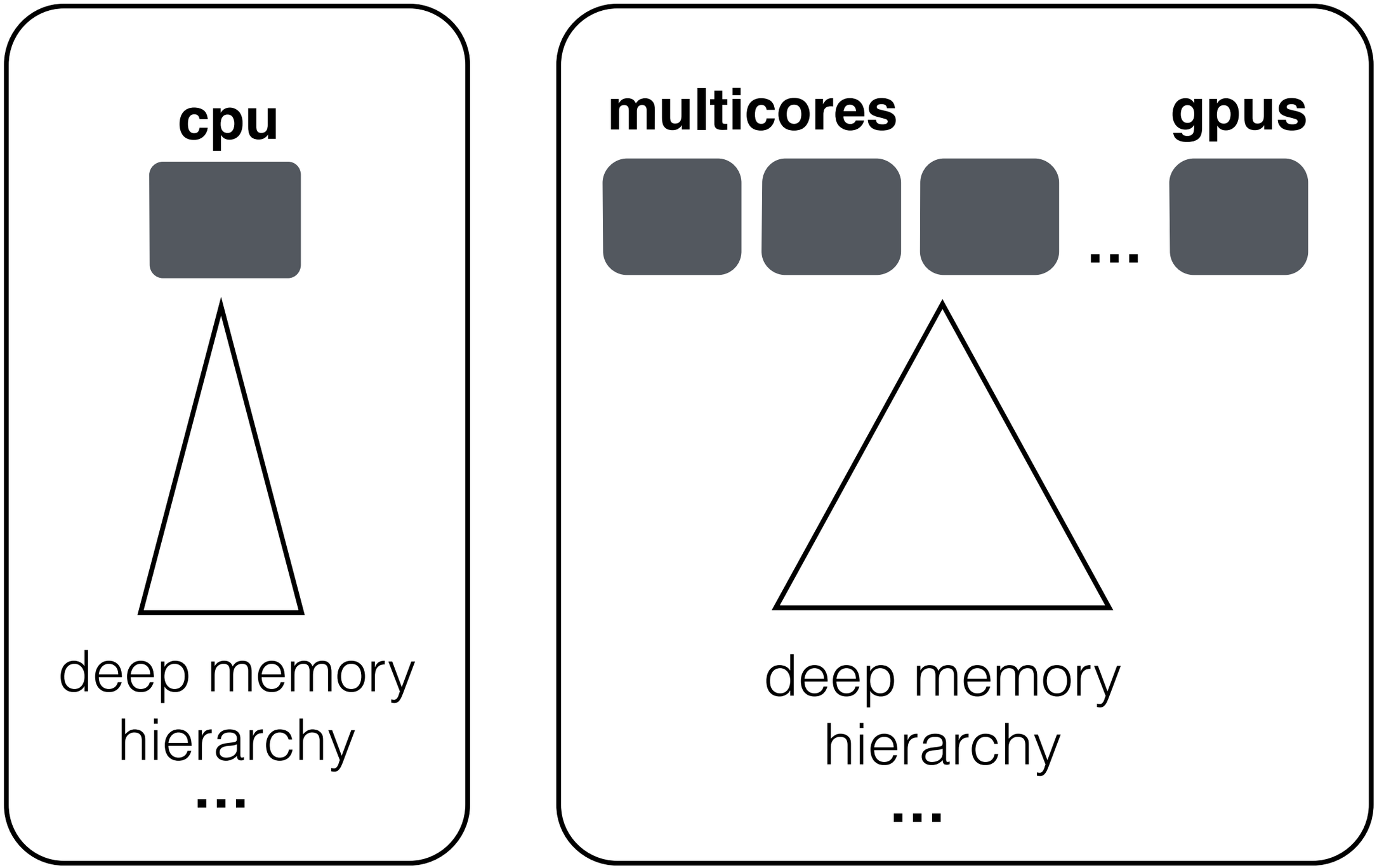


2. can predict data accesses

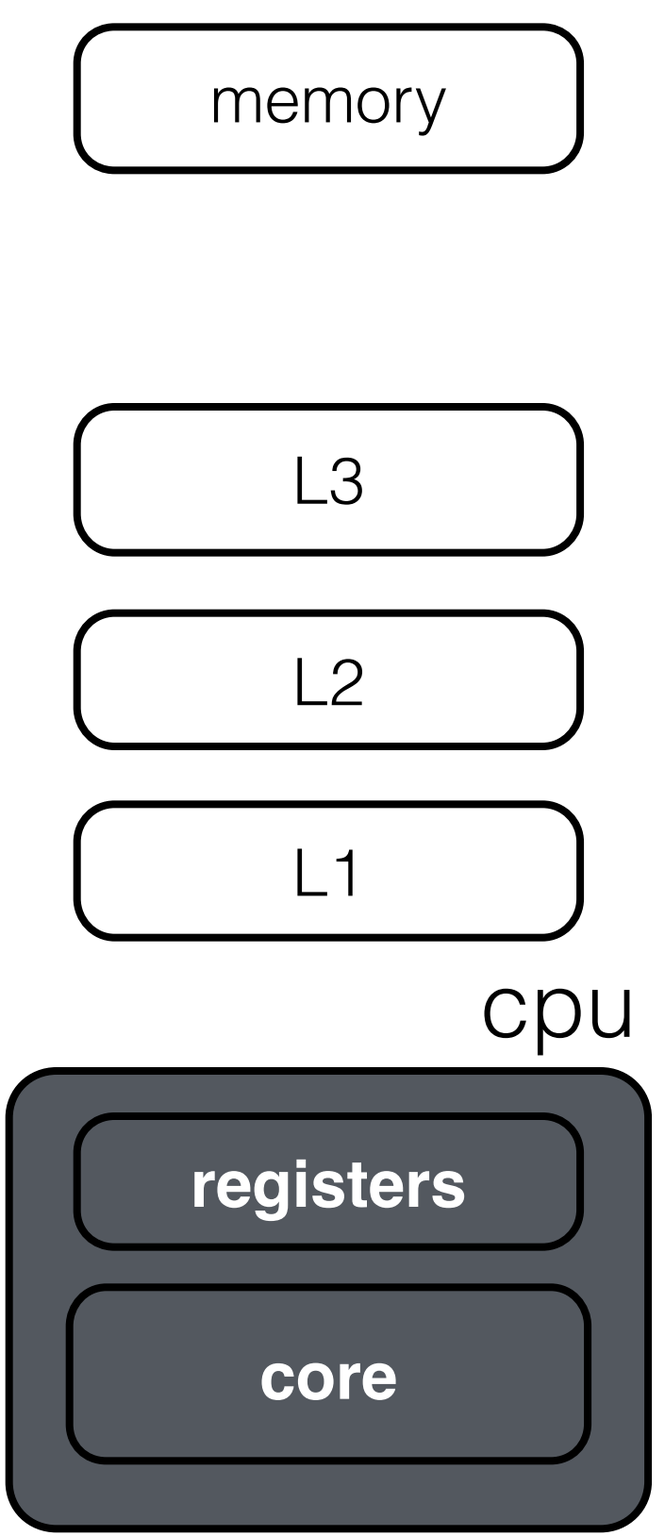
3. can do the same task on >1 data

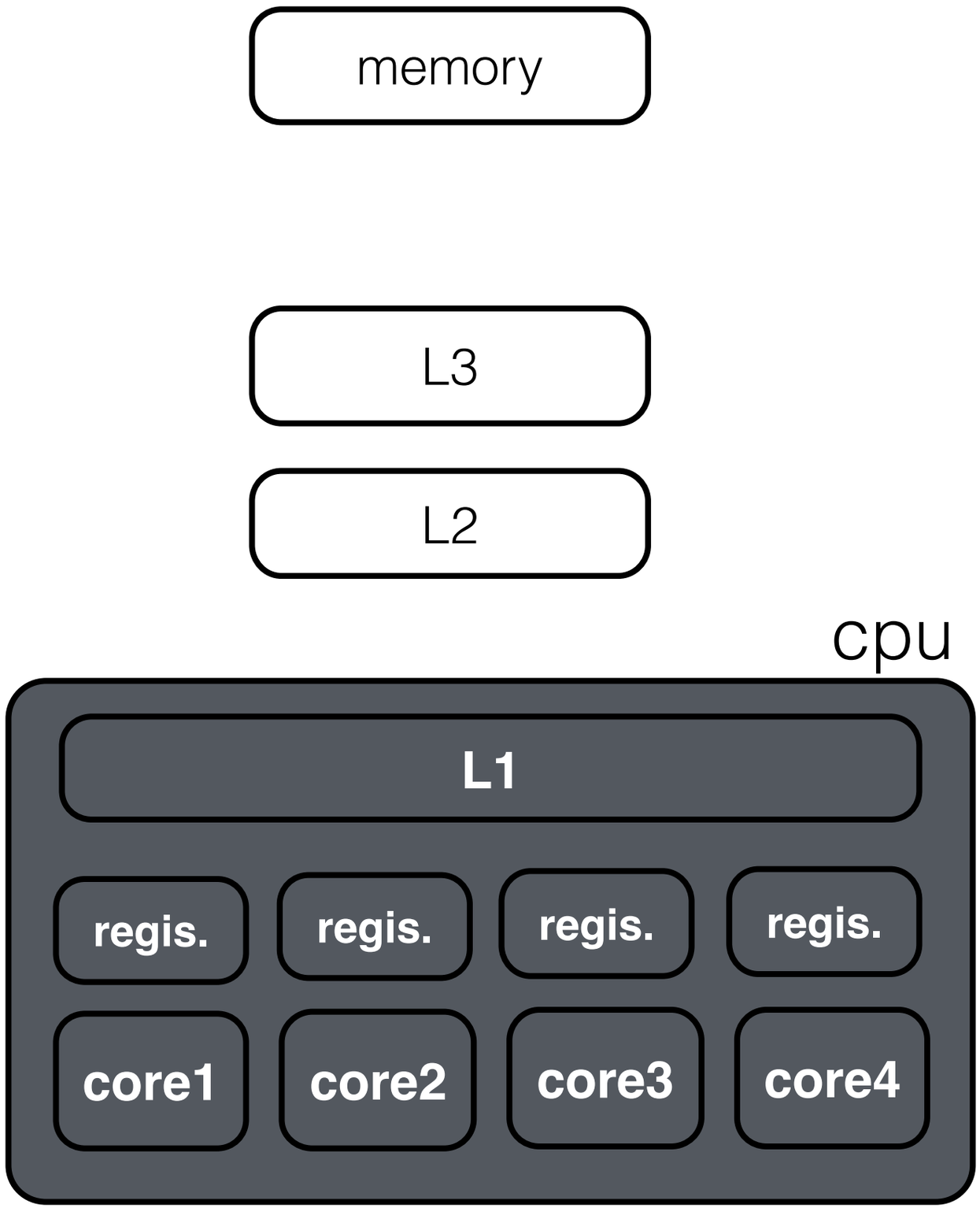
1. can do >1 tasks at the same time

cpu

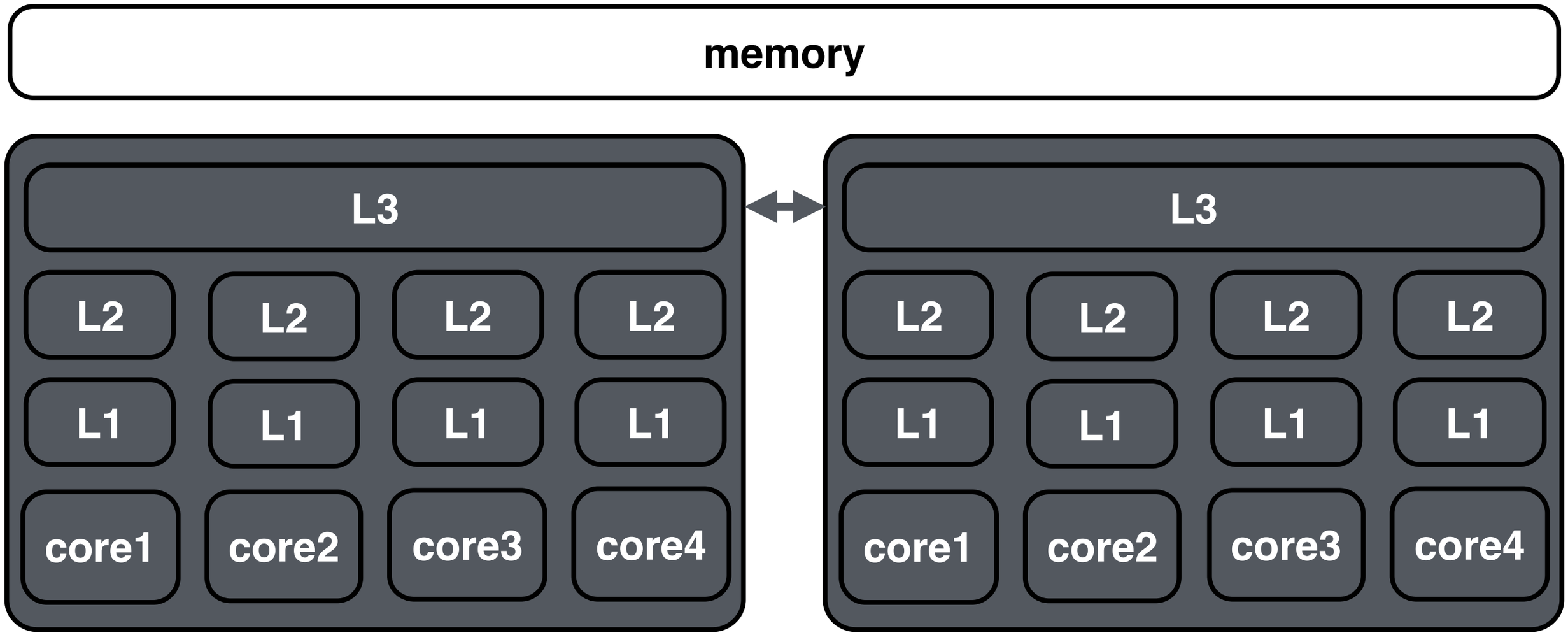

cpu
deep memory
hierarchy
...
multicores
gpus
...
deep memory
hierarchy
...

# from single core to multi-core

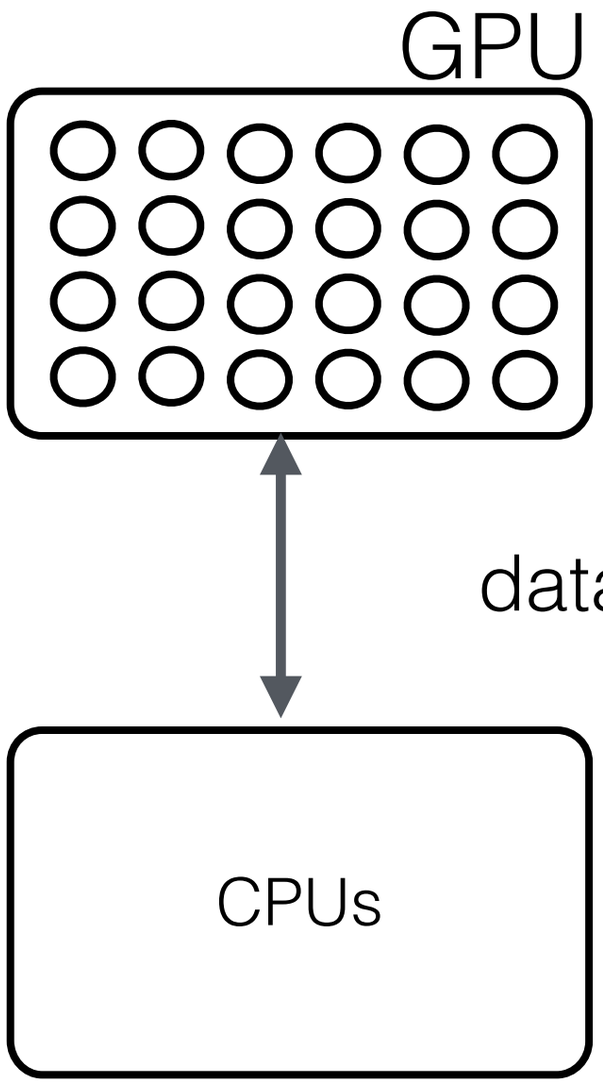


# from multi-core to NUMA
## data placement becomes crucial

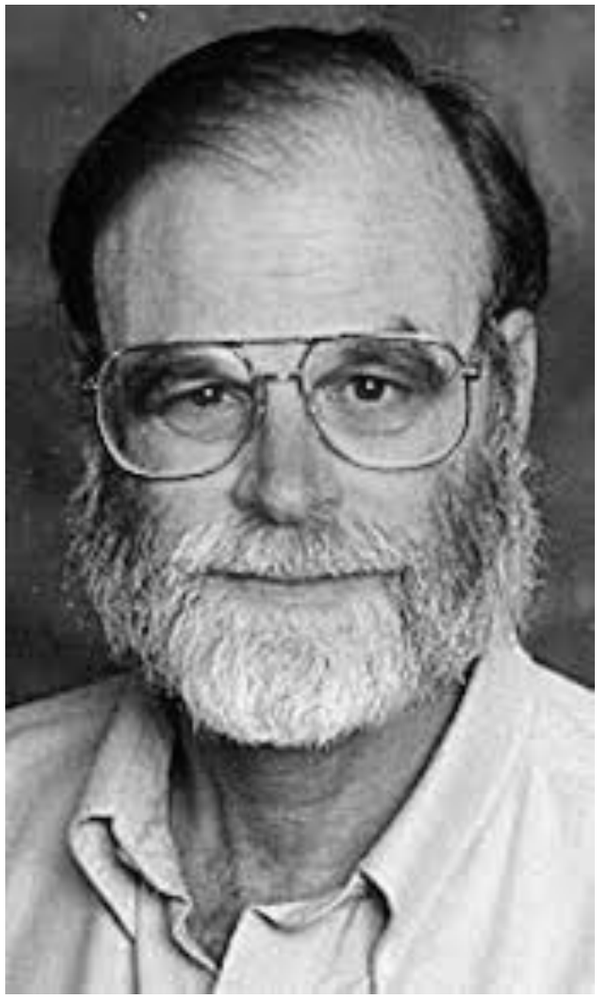



**ATraPos: Adaptive transaction processing on hardware Islands**
Danica Porobic, Erietta Liarou, Pinar Tözün, Anastasia Ailamaki
International Conference on Data Engineering (ICDE), 2014




many “small” cores (1000s)
subsets of cores work on same task
so branches are again problematic

data transfer may be expensive



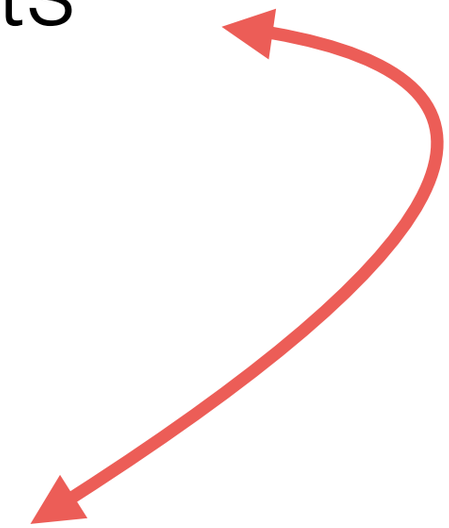
Jim Gray, IBM, Tandem, DEC, Microsoft
ACM Turing award
ACM SIGMOD Edgar F. Codd Inovations award

| | |
|---|---|
| **100Kx** disk | **Pluto** 2 years |
| **100x** memory | **New York** 1.5 hours |
| **10x** on board cache | **this building** 10 min |
| **2x** on chip cache | **this room** 1 min |
| registers | **my head** ~0 |

**many more levels and latency differences**

utilization

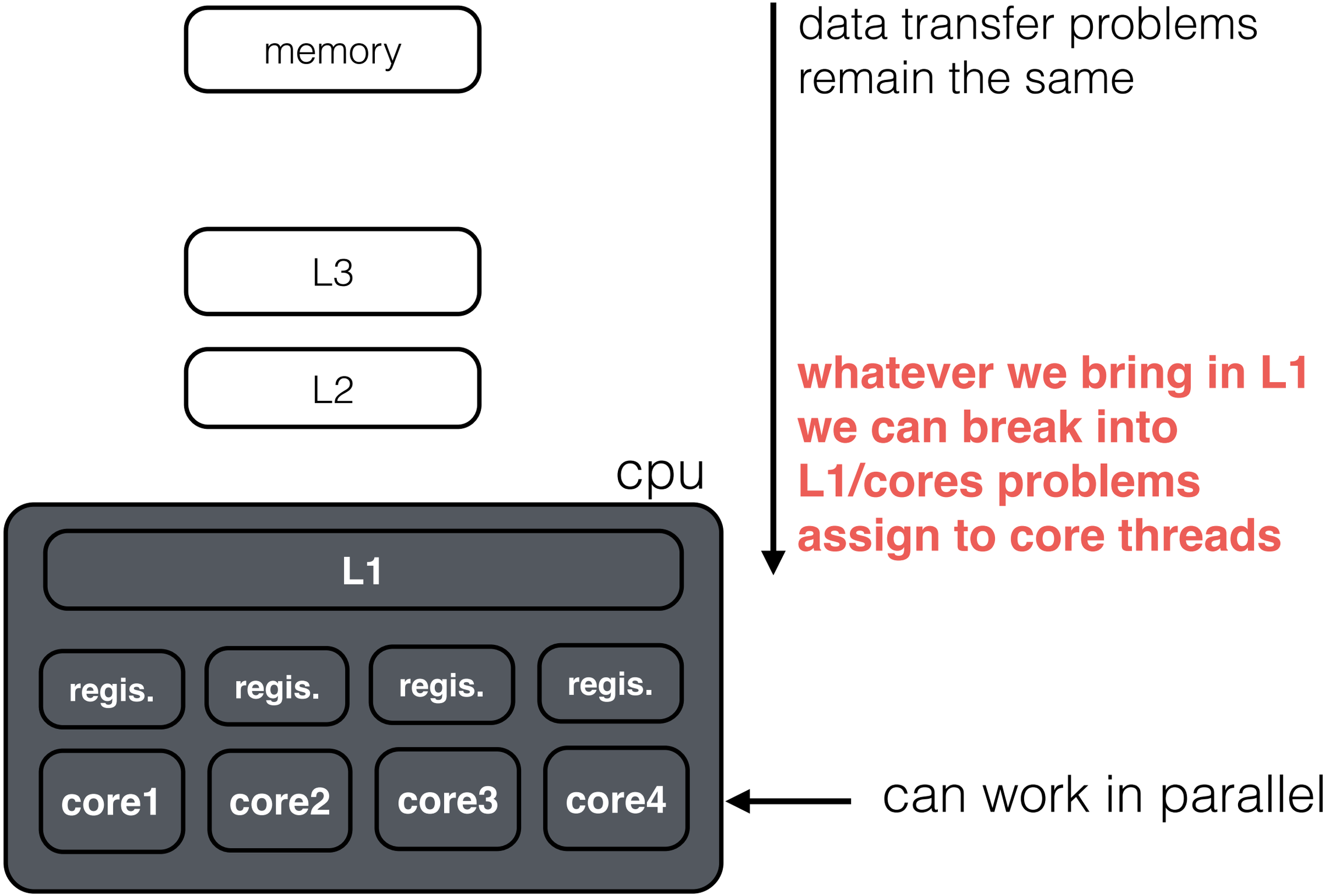
energy side-effects



# challenge

how do we keep all CPUs/cores at 100%

**Dynamic fine-grained scheduling for energy-efficient main-memory queries**
Iraklis Psaroudakis et al
International Workshop on Data Management on New Hardware (**DAMON**), 2014




data transfer problems remain the same

**whatever we bring in L1 we can break into L1/cores problems assign to core threads**

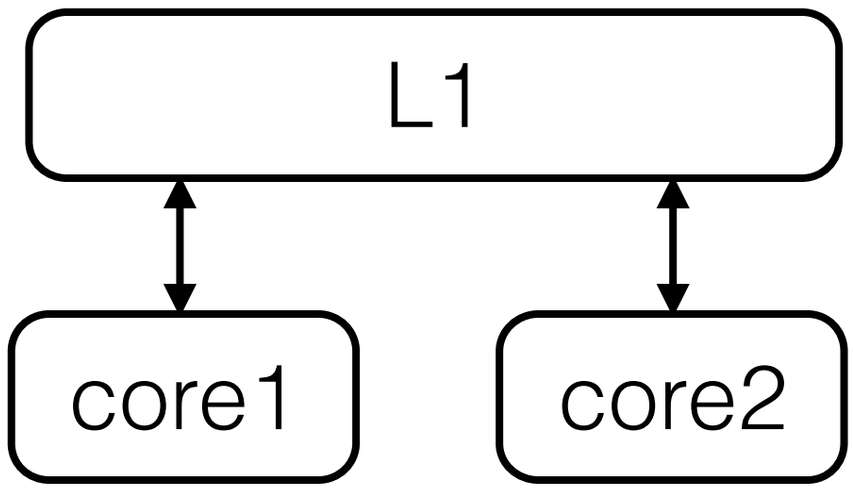
can work in parallel

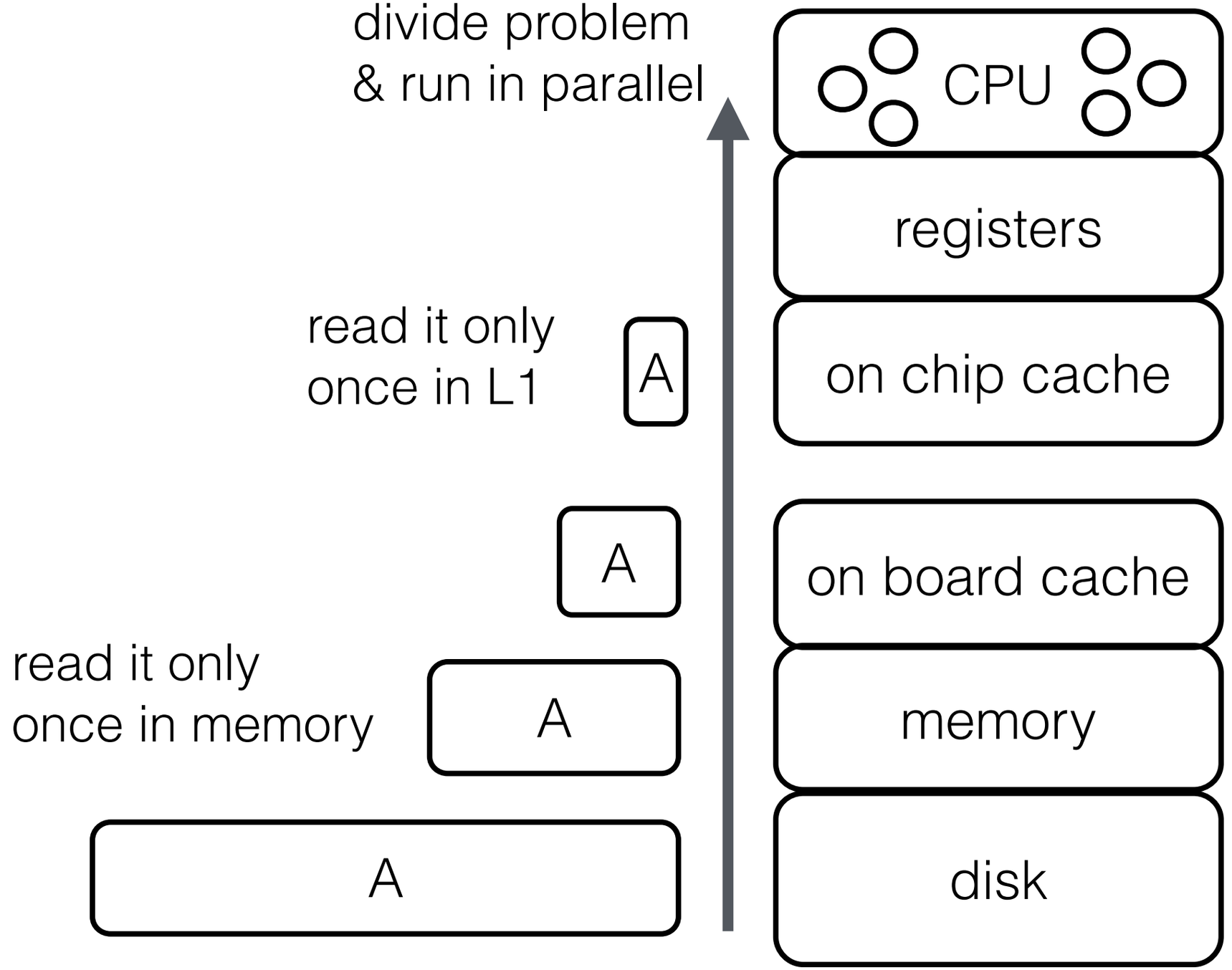
**operator**: average

memory

56, 34, 12, 1, 87, 22, 98, 49, 7, 12, …

data size=L1 size x 10




divide problem
& run in parallel
CPU
registers
on chip cache
read it only
once in L1
A
A
on board cache
read it only
once in memory
A
memory
A
disk

## loop fusion

watch out for *data locality*

```
for(i=0;i<1000;i++)
    min = a[i]<min ? a[i] : min

for(i=0;i<1000;i++)
    max = a[i]>max ? a[i] : max
```

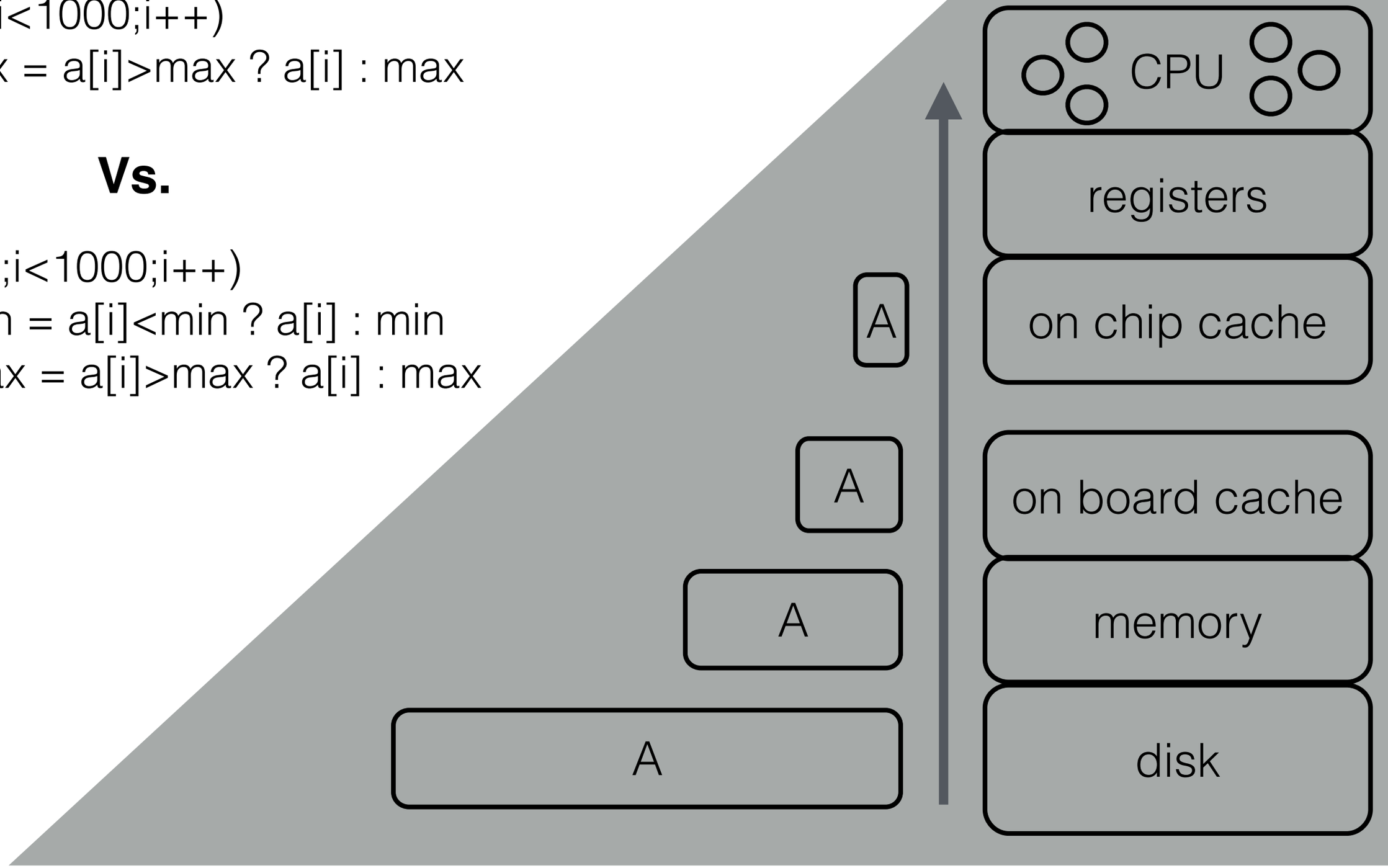
```
for(i=0;i<1000;i++)
    min = a[i]<min ? a[i] : min
    max = a[i]>max ? a[i] : max
```

```
for(i=0;i<1000;i++)
      min = a[i]<min ? a[i] : min

for(i=0;i<1000;i++)
      max = a[i]>max ? a[i] : max
```

**Vs.**

```
 for(i=0;i<1000;i++)
       min = a[i]<min ? a[i] : min
       max = a[i]>max ? a[i] : max
```



## loop fission

watch out for data locality

```
for(i=0;i<1000;i++)
     min = a[i]<min ? a[i] : min
     max = b[i]>max ? b[i] : max
```

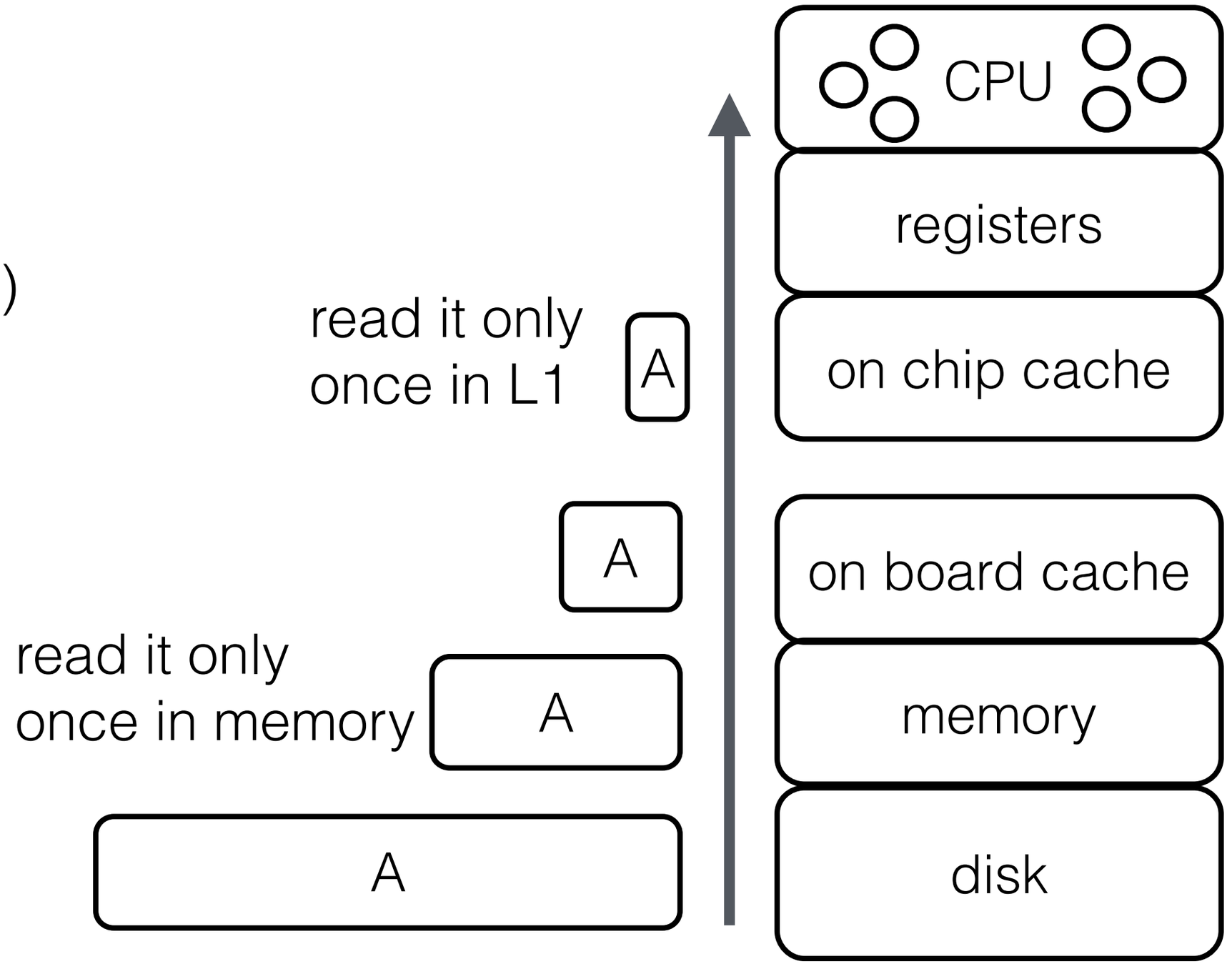
```
for(i=0;i<1000;i++)
     min = a[i]<min ? a[i] : min

for(i=0;i<1000;i++)
     max = b[i]>max ? b[i] : max
```

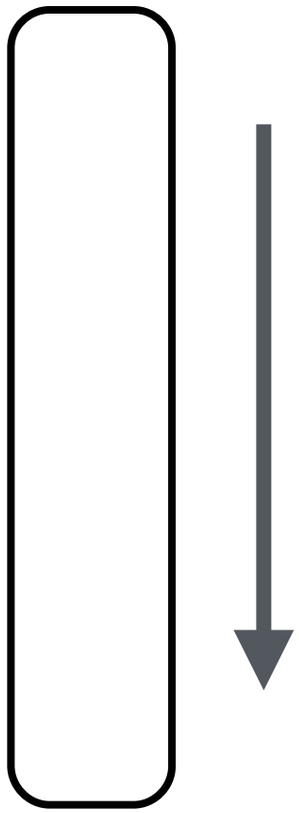
# vectorization

process one block of data at a time

**select** min(A), max(A)
**from** R
**where** A<10



**full scan:** for every tuple check if the value satisfies the predicate and if it does remember the position of the tuple

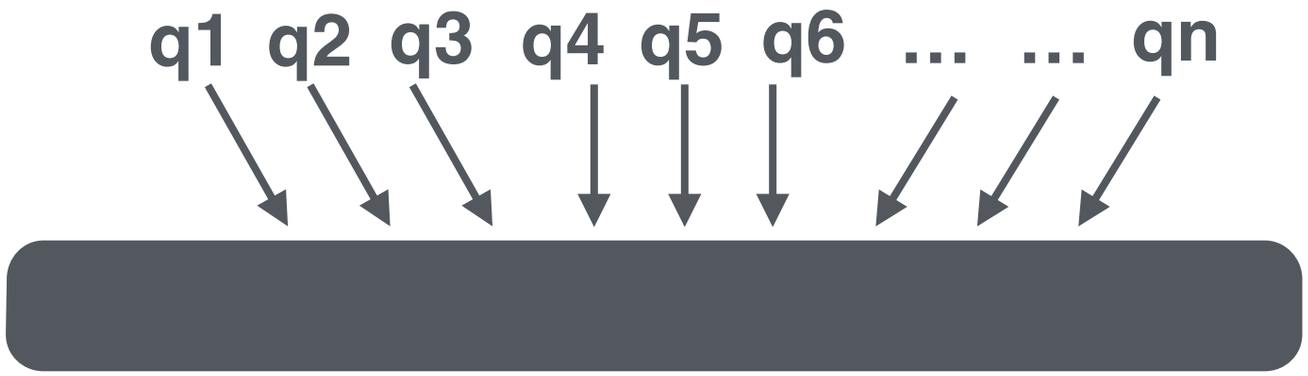


```
select(input,low,high,inclusiveLow,inclusiveHigh)
1: int *output = new array(sizeOf(input))
2:  for (i=0;i<tuples;i++,input++)
3:        if *input>=low && *input<high
4:          *output++=i
5:  return output
```

sequential access pattern=good for CPU + memory hierarchy
(next class more about why this is true)

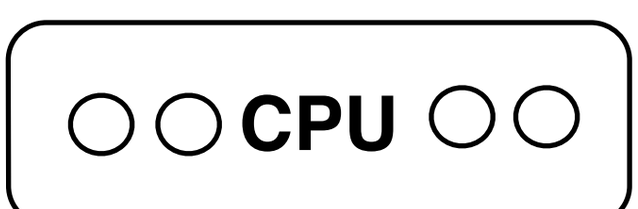
**what if we have >>1 queries arriving in parallel?**

how can we keep all CPUs busy to 100%
& minimize data movement?

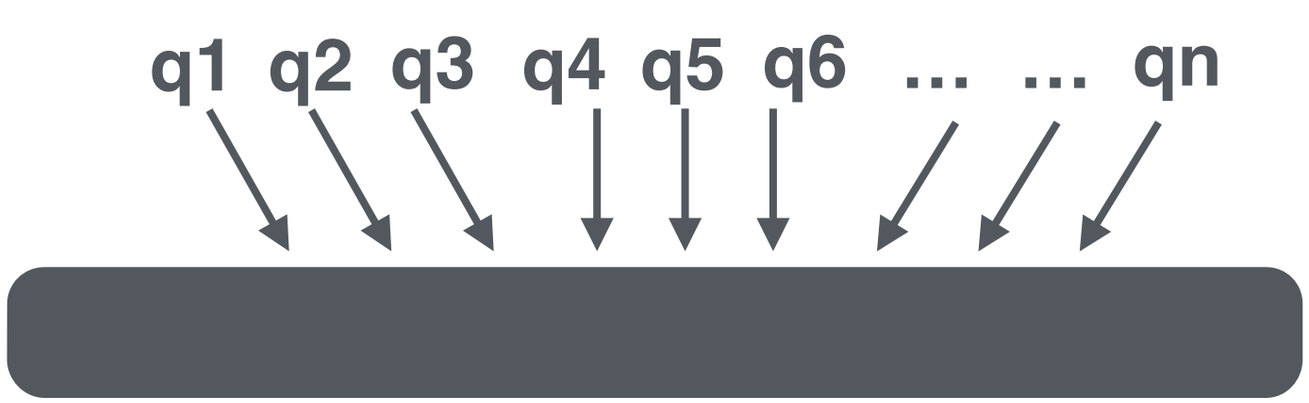


N queries/selects in parallel on the same column
1) cost (L1 misses) for plain scan
2) devise shared scan approach
3) cost (L1 misses) for shared scan

corner cases:
what if queries do not arrive at the same time?
what if some queries are faster than others?
is there a limit to the number of queries in a shared a scan?








(assume simplified memory hierarchy)

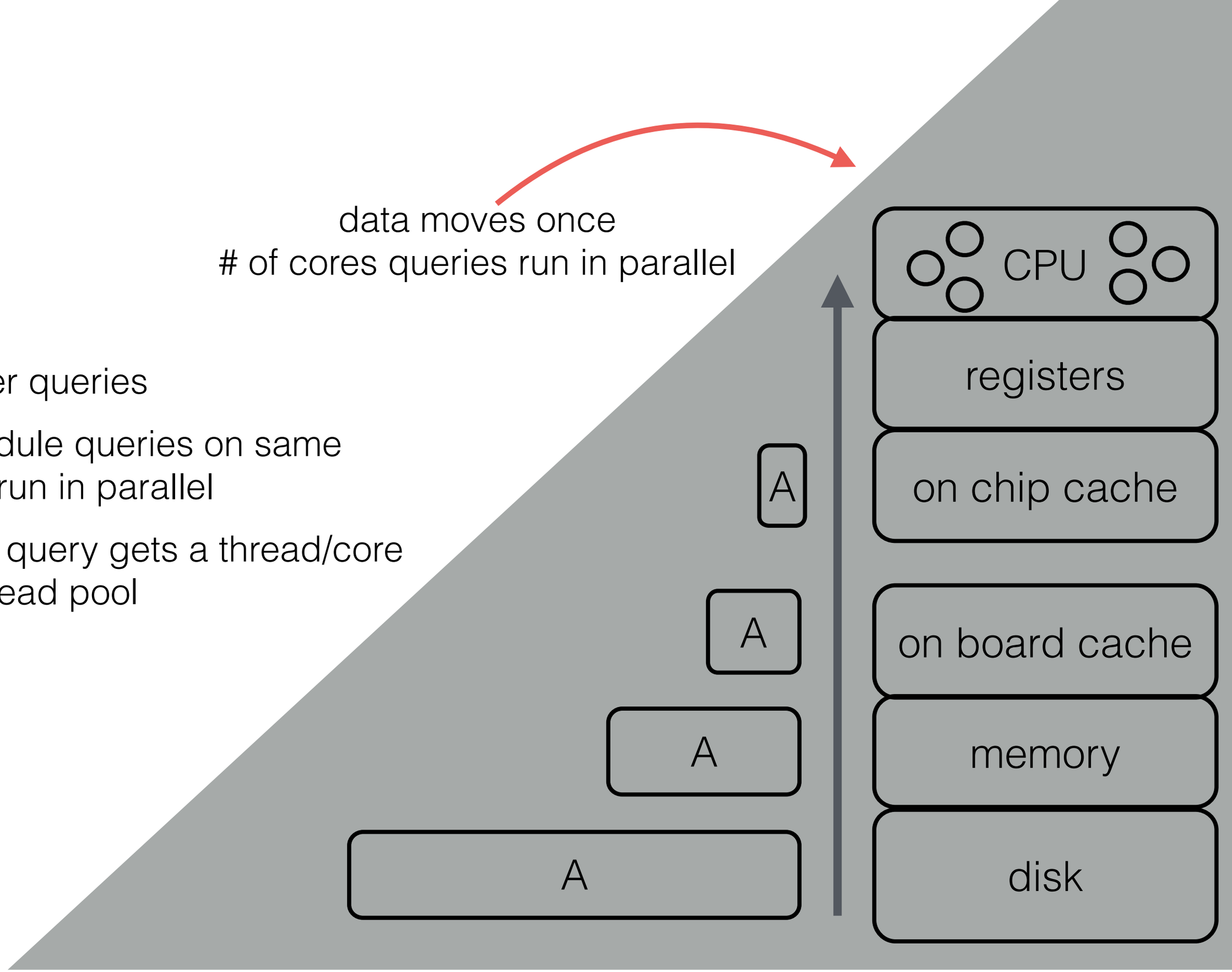
Column >L1, Column < L2, L1 block = L2 block = block bytes, Column = C blocks
CPU can read directly from level 1 only

data moves once
# of cores queries run in parallel

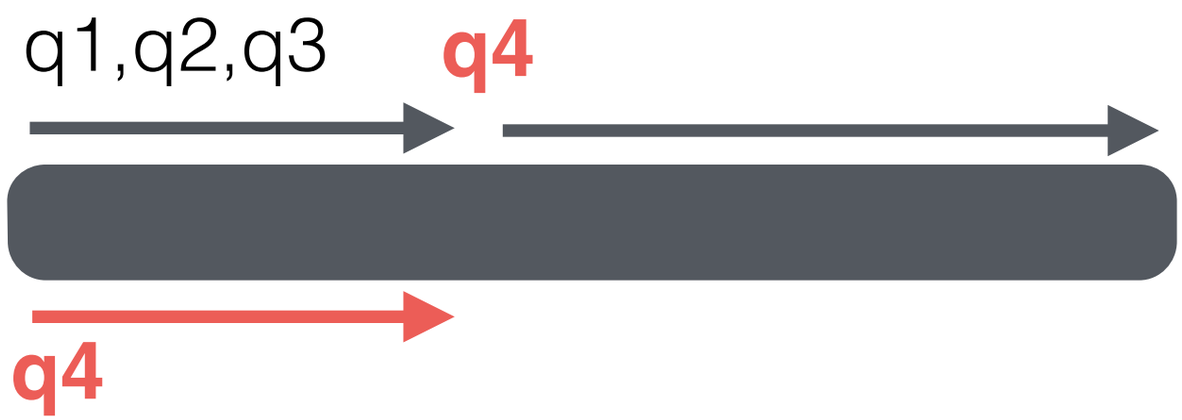
1) gather queries

2) schedule queries on same data to run in parallel

3) each query gets a thread/core from thread pool



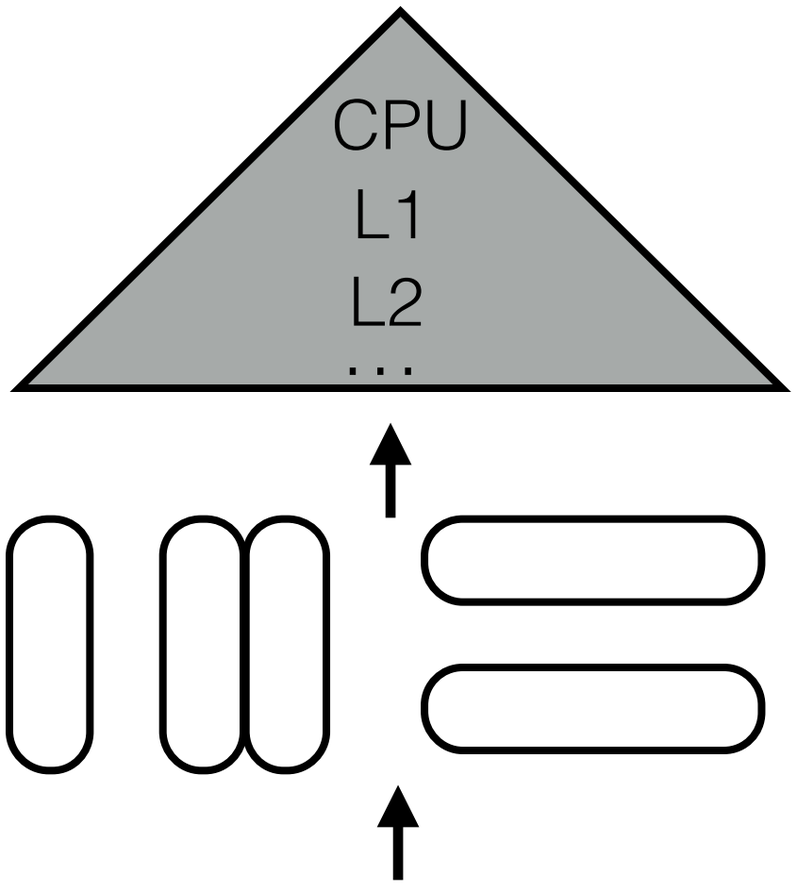
**attach** queries arriving asynchronously
**elevate** queries that are slow




## what to do for milestone 3?

assume you have all queries
minimize data movement, try to utilize CPUs 100%
ideally (within reason) shared scan scales with # of queries and # of CPUs
numa: straightforward partitioning is ok

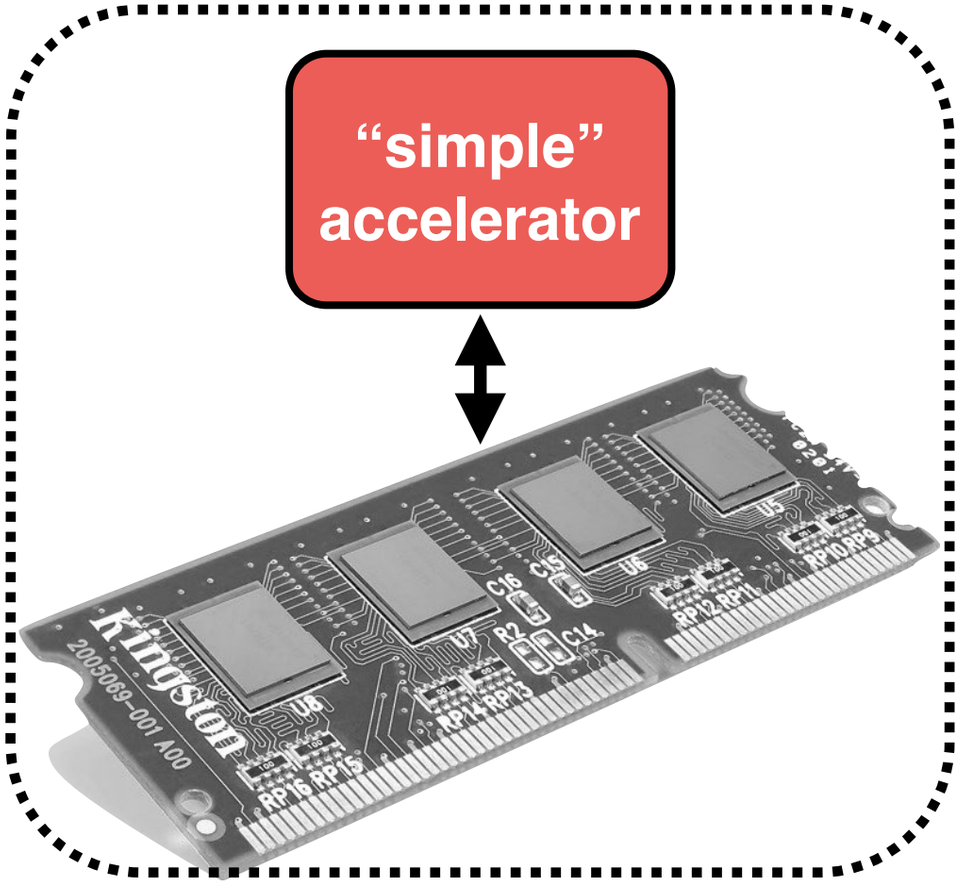
bonus: queries arriving asynchronously




hardware/software co-design

energy - speed

**Navigating big data with high-throughput, energy-efficient data partitioning.**
L. Wu, R. J. Barker, M. A. Kim, K. A. Ross
International Symposium on Computer Architecture, 2013




**Meet the walkers: Accelerating index traversals for in-memory databases.**
O. Koçberber, B. Grot, J. Picorel, B. Falsafi,
K. T. Lim, P. Ranganathan
International Symposium on Microarchitecture, 2013

**Beyond the Wall: Near-Data Processing for Databases**
S. Xi, O. Babarinsa, M. Athanassoulis, S. Idreos.
International Workshop on Data Management
on New Hardware, 2015

**textbook:** Chapters 8,9

**Cooperative Scans: Dynamic Bandwidth Sharing in a DBMS**
Marcin Zukowski, Sándor Héman, Niels Nes, Peter A. Boncz
Very large Databases Conference (**VLDB**), 2007

**Morsel-driven parallelism: a NUMA-aware query evaluation framework for the many-core age**
Viktor Leis, Peter A. Boncz, Alfons Kemper, and Thomas Neumann
ACM **SIGMOD** International Conference on Management of Data, 2014

**next:** fast scans & modern hardware 2.0

# fast scans 1.0

# DATA SYSTEMS

prof. Stratos Idreos