# Computer Architectures

Compute as you sew

Florent Chuffart – INSERM/IAB/EpiMed

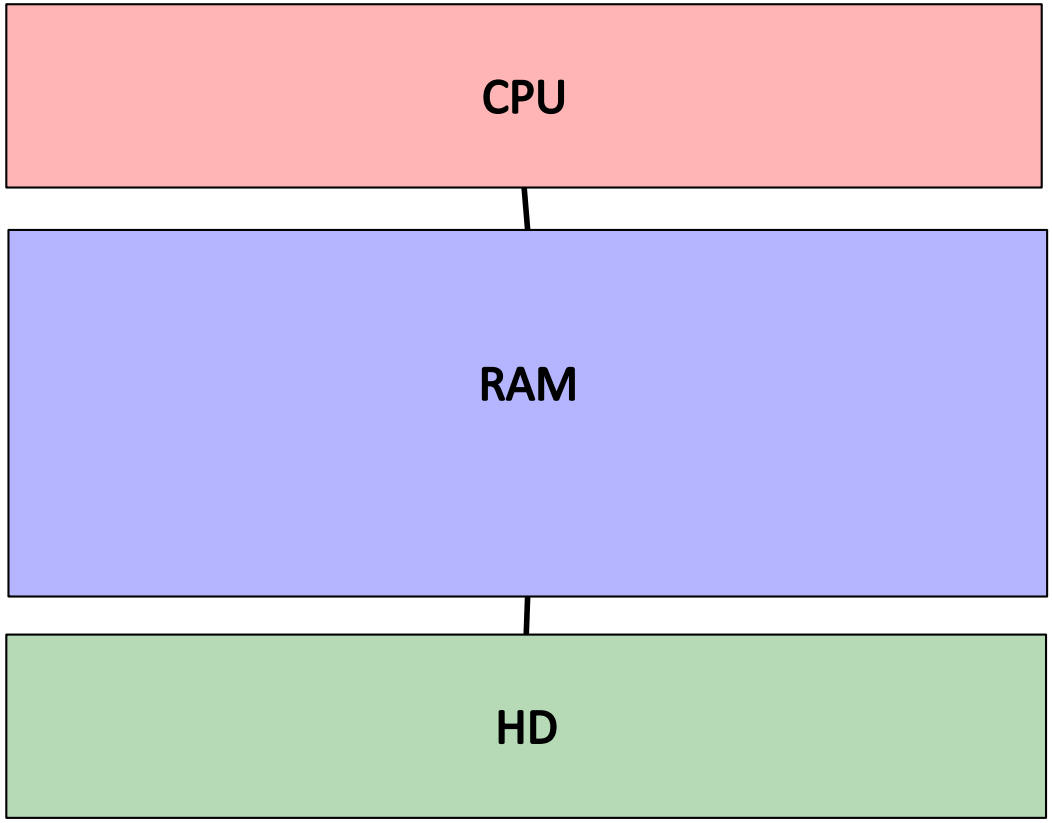
# Plan

- Computer Architectures
- 2 Parallel Programming Paradigms

# Basic Computer Architecture

- CPU
- RAM
- HD

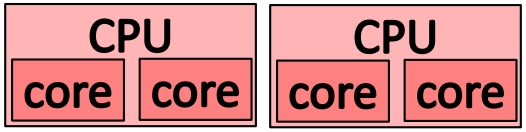
# Basic Model



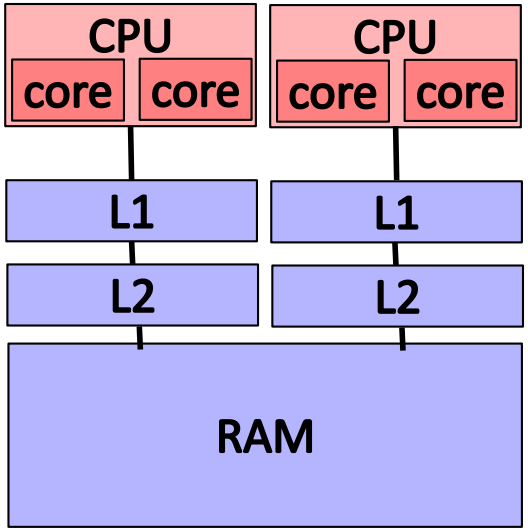
# Advanced Computer Architecture

- Hyper Threading
- Multicore
- Multiple CPUs
- Many Cache Level
- Non Uniform Memory Access (NUMA)

# Multiple CPUs, Multicore



# Cache levels

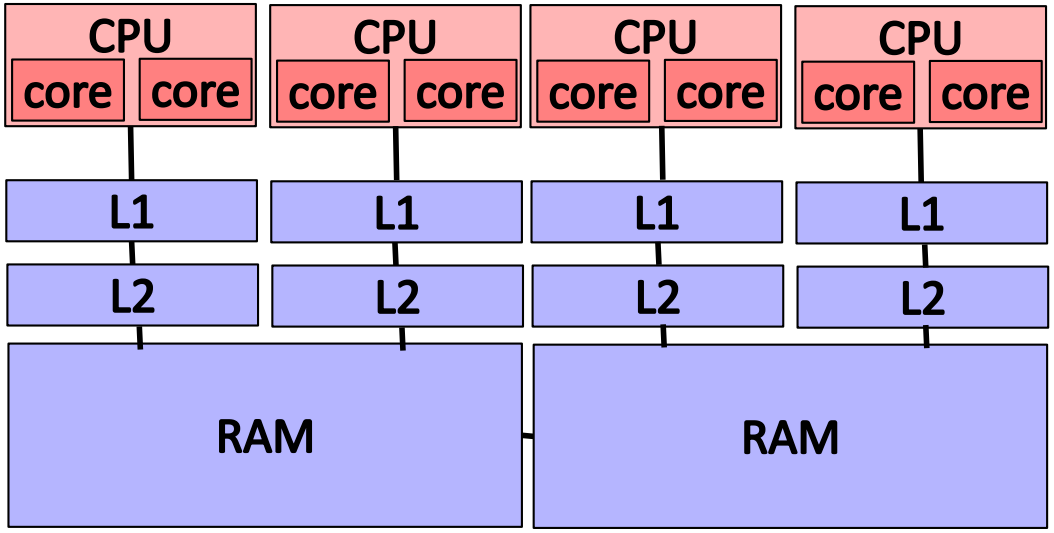



super fast access

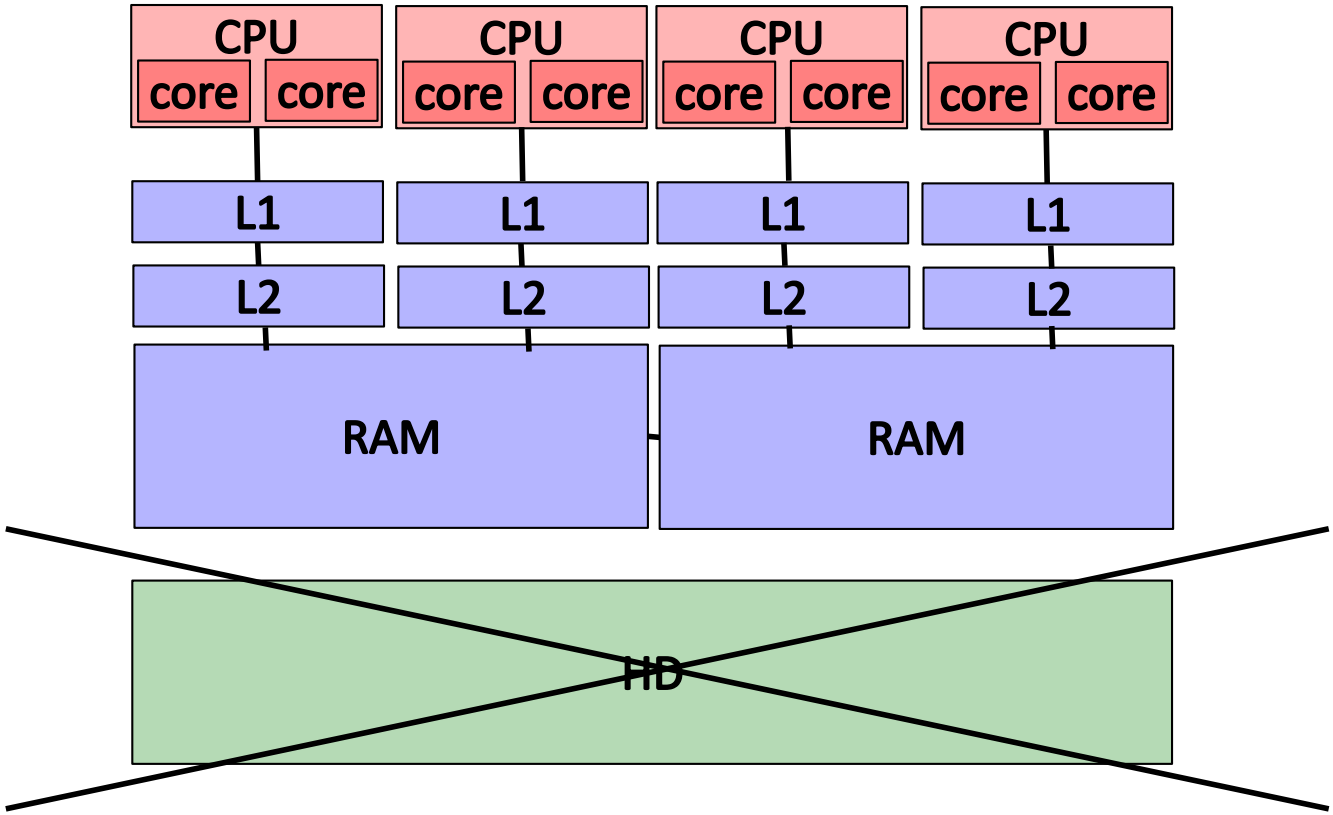
fast access

less fast access

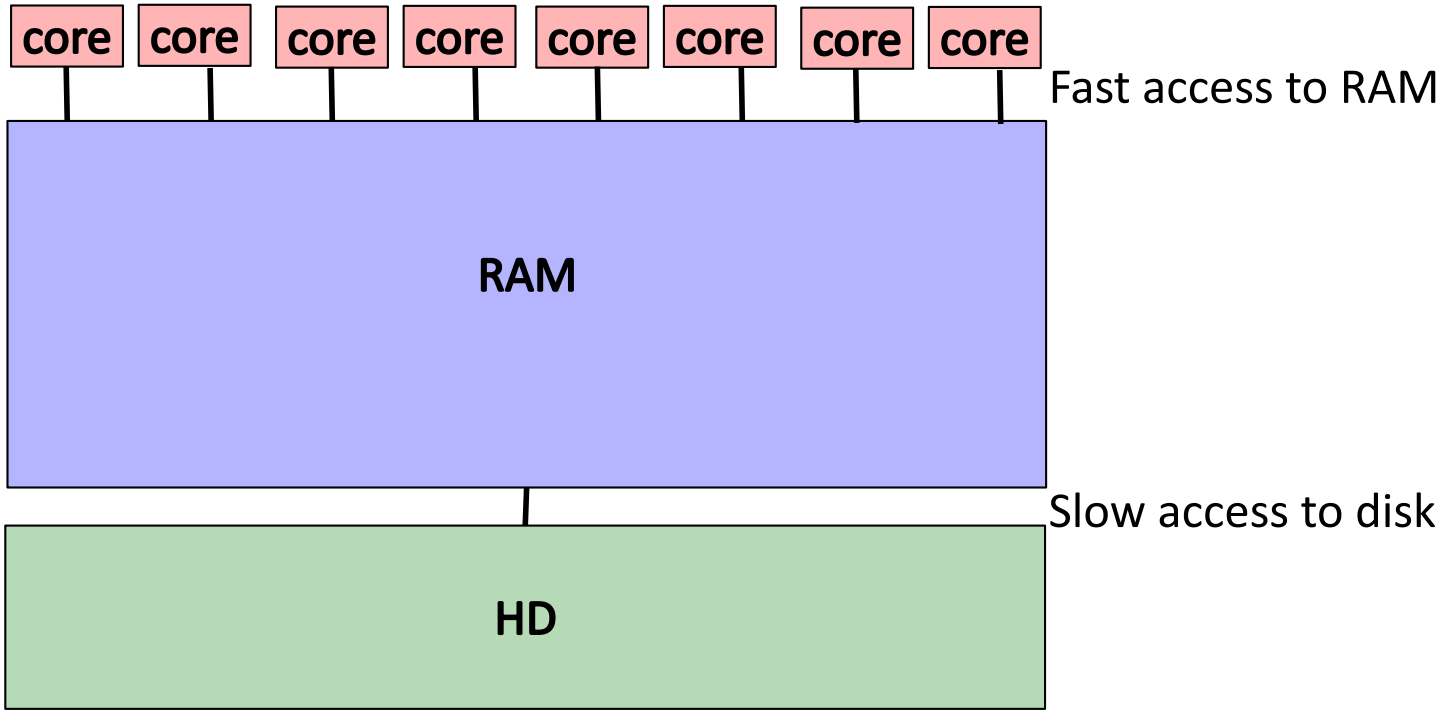
# Cache levels and NUMA



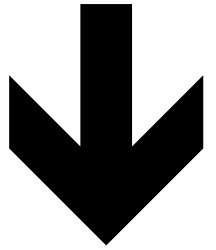
# No Hard Drive in a Near Future



# Take Home Model

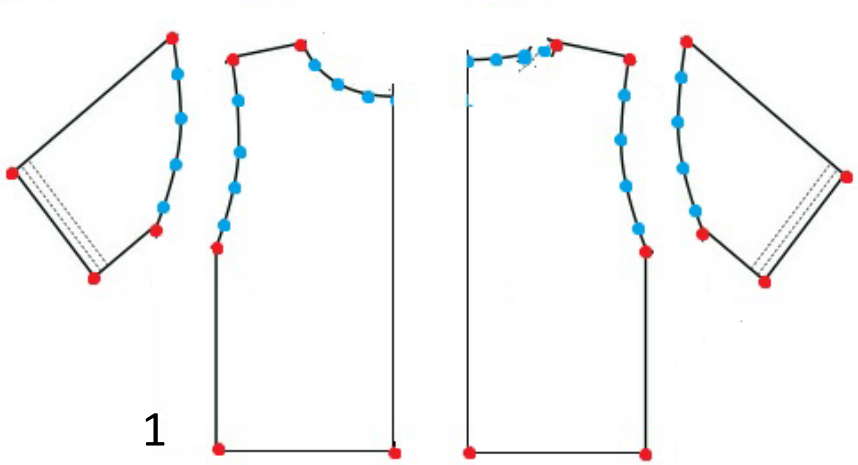


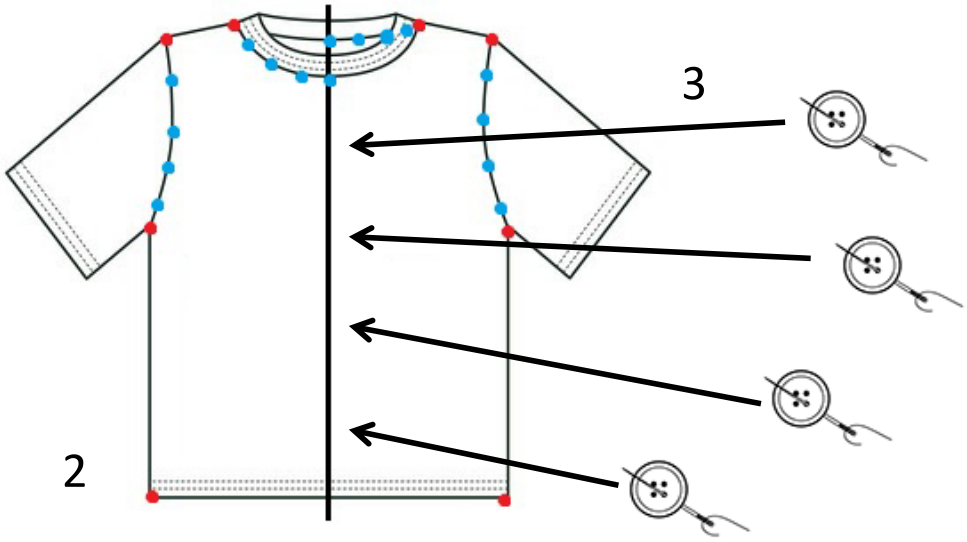
# Many Cores



# Parallel Programming

# Example of parallelism




1 – parallel (build of many pieces)
2 – sequential (merge pieces)
3 – parallel (sew buttons)

# 2 Parallel Programming Paradigms

- Distributed Memory

- Shared Memory

# Distributed Memory

# Distributed Memory

- 3 samples (sample1, sample2, sample3)
- 2 tasks (align and count)

# Sequential

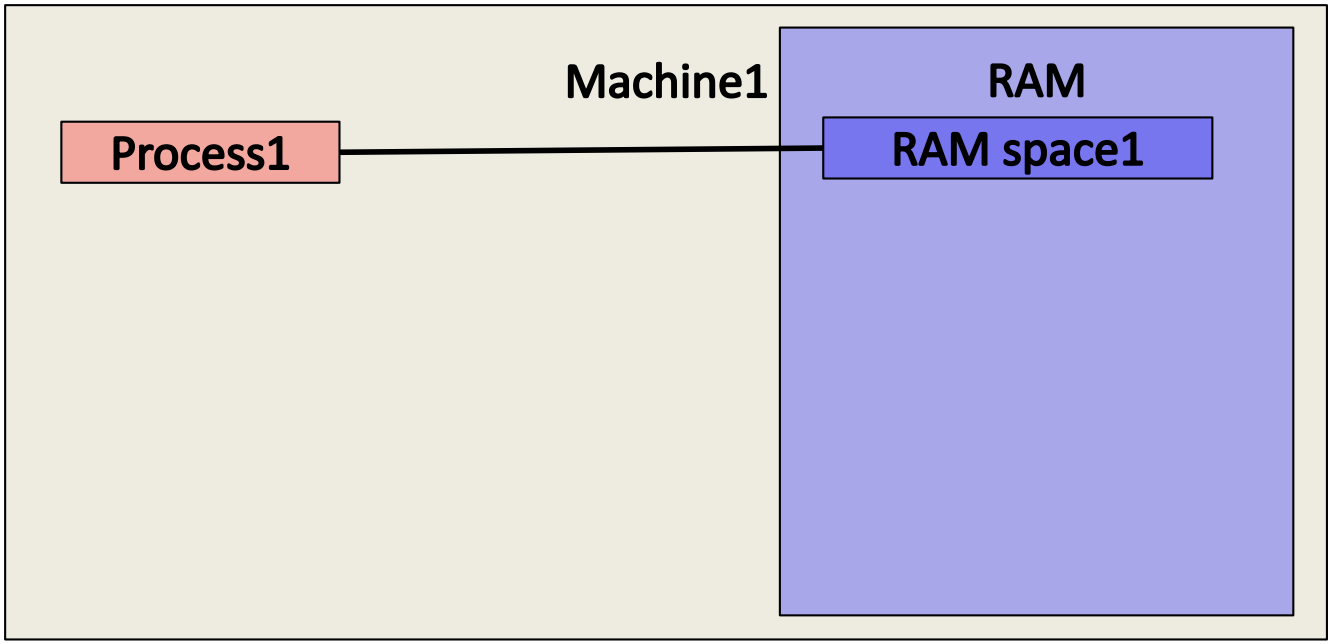
- 3 samples (sample1, sample2, sample3)
- 2 tasks (align and count)

```
align sample1 | count > tab1.tab ;
align sample2 | count > tab2.tab ;
align sample3 | count > tab3.tab ;
```

# Sequential

- 3 samples (sample1, sample2, sample3)
- 2 tasks (align and count)

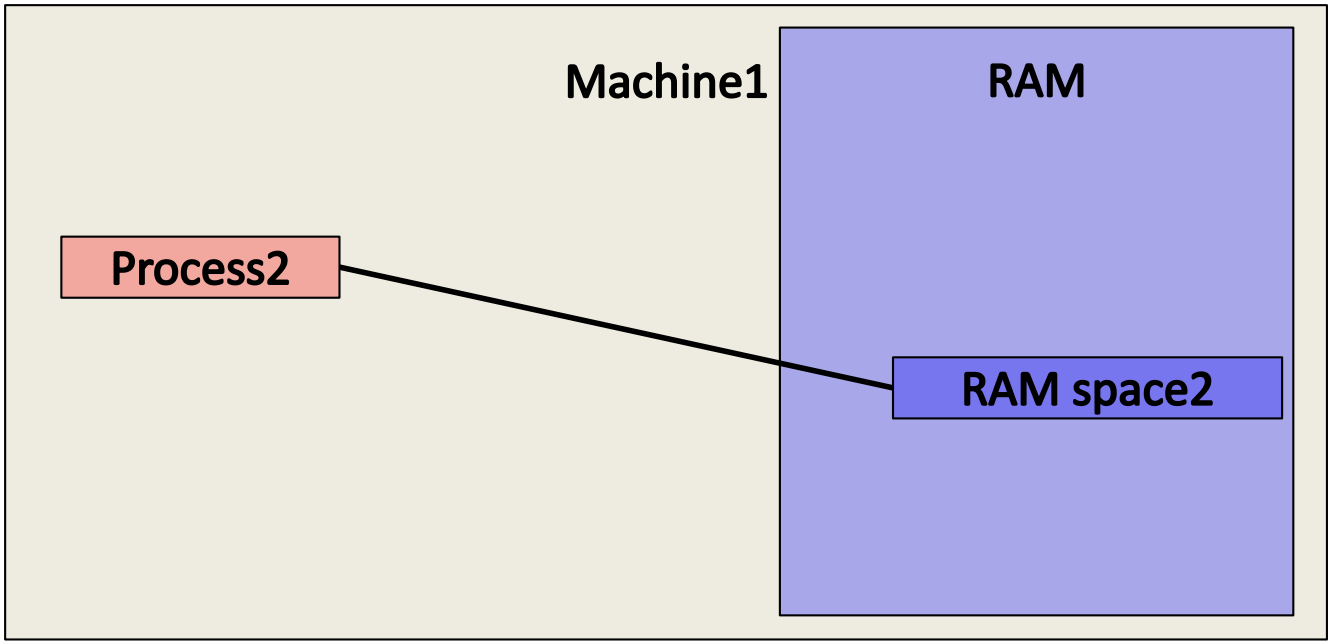
```
align sample1 | count > tab1.tab ;
align sample2 | count > tab2.tab ;
align sample3 | count > tab3.tab ;
```



# Sequential

- 3 samples (sample1, sample2, sample3)
- 2 tasks (align and count)

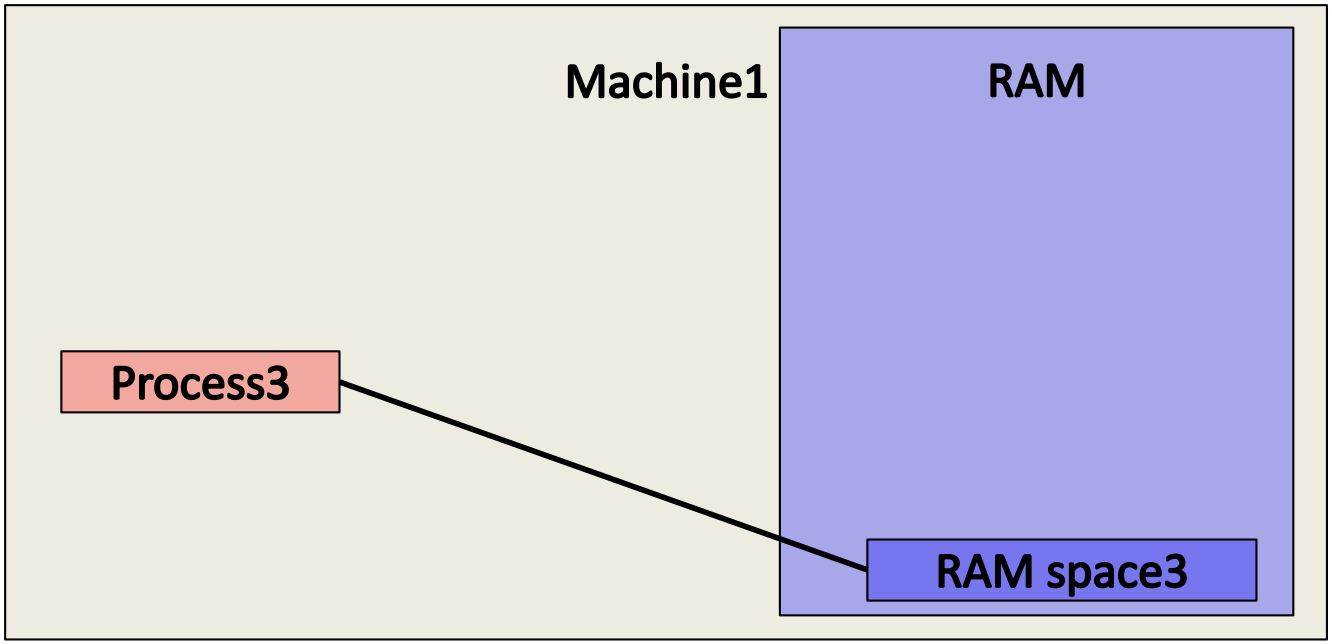
```
align sample1 | count > tab1.tab ;
align sample2 | count > tab2.tab ;
align sample3 | count > tab3.tab ;
```



# Sequential

- 3 samples (sample1, sample2, sample3)
- 2 tasks (align and count)

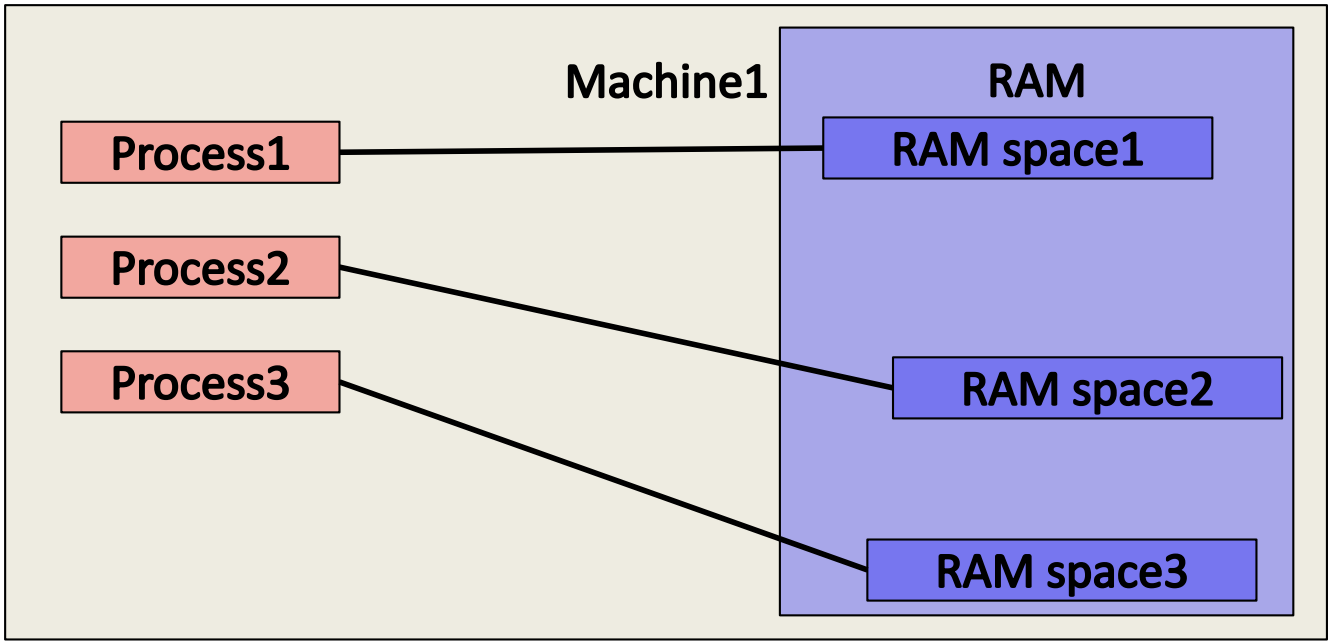
```
align sample1 | count > tab1.tab ;
align sample2 | count > tab2.tab ;
align sample3 | count > tab3.tab ;
```



# Parallel on 1 Machine

- 3 samples (sample1, sample2, sample3)
- 2 tasks (align and count)

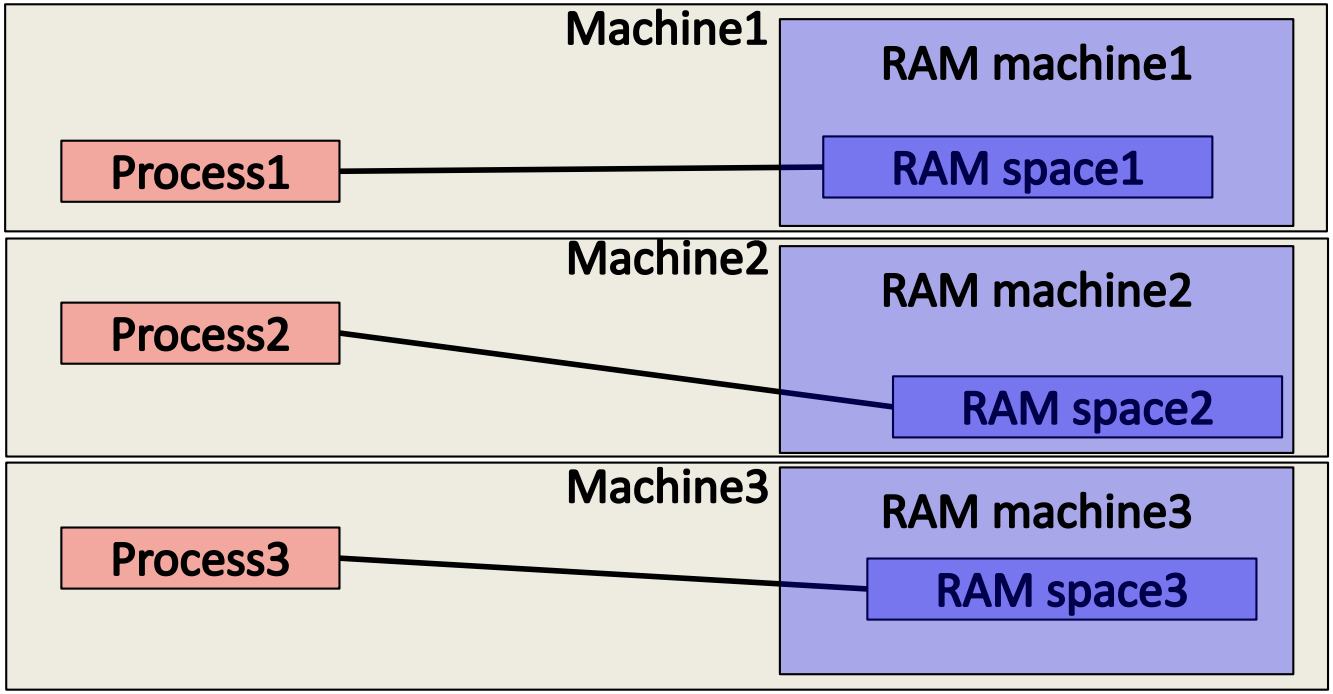
```
align sample1 | count > tab1.tab &
align sample2 | count > tab2.tab &
align sample3 | count > tab3.tab &
```



# Parallel on a Cluster (Ciment)

- 3 samples (sample1, sample2, sample3)
- 2 tasks (align and count)

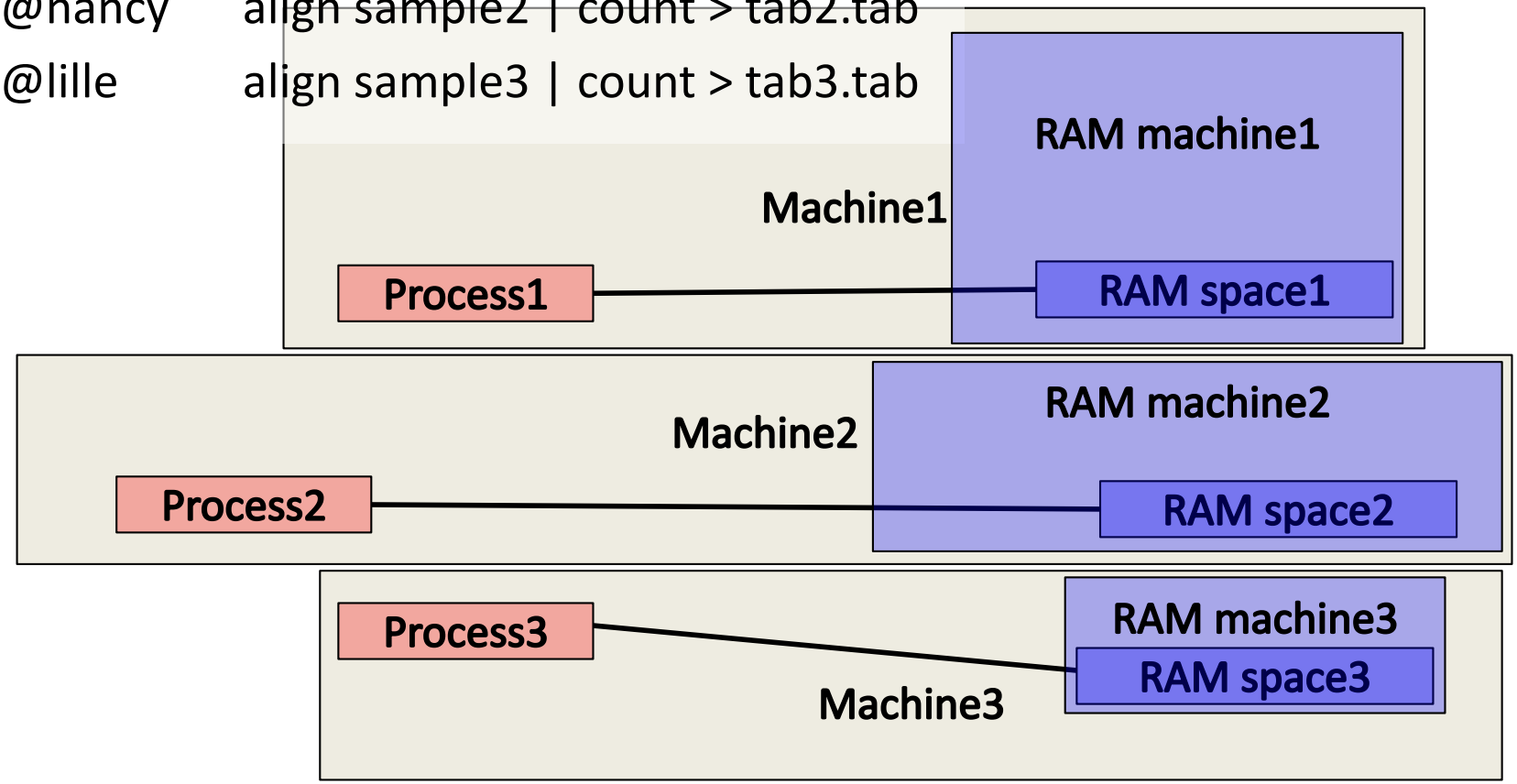
```
ssh mach1@ciment    align sample1 | count > tab1.tab
ssh mach2@ciment    align sample2 | count > tab2.tab
ssh mach3@ciment    align sample3 | count > tab3.tab
```



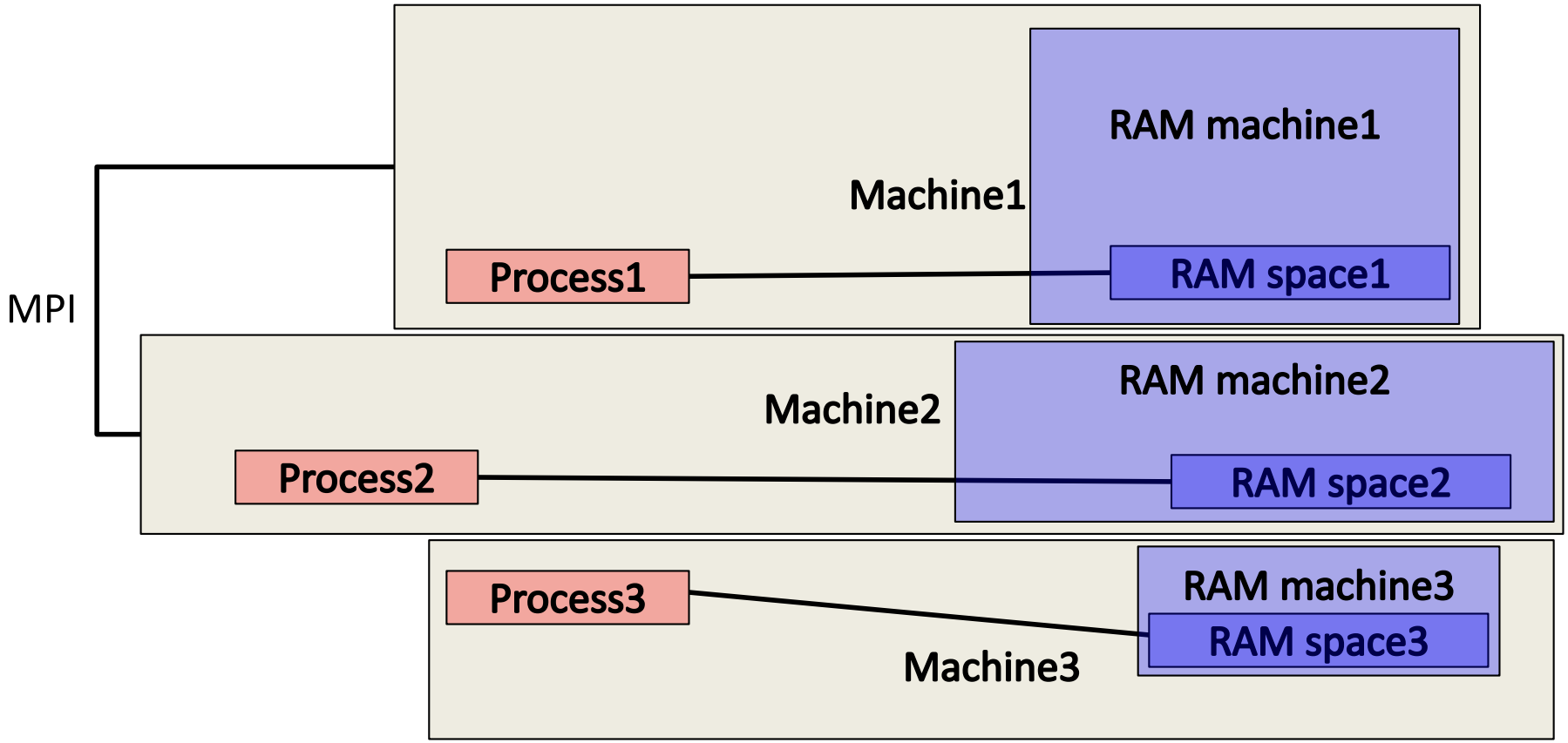
# Parallel on a Grid (Grid5000)

- 3 samples (sample1, sample2, sample3)
- 2 tasks (align and count)

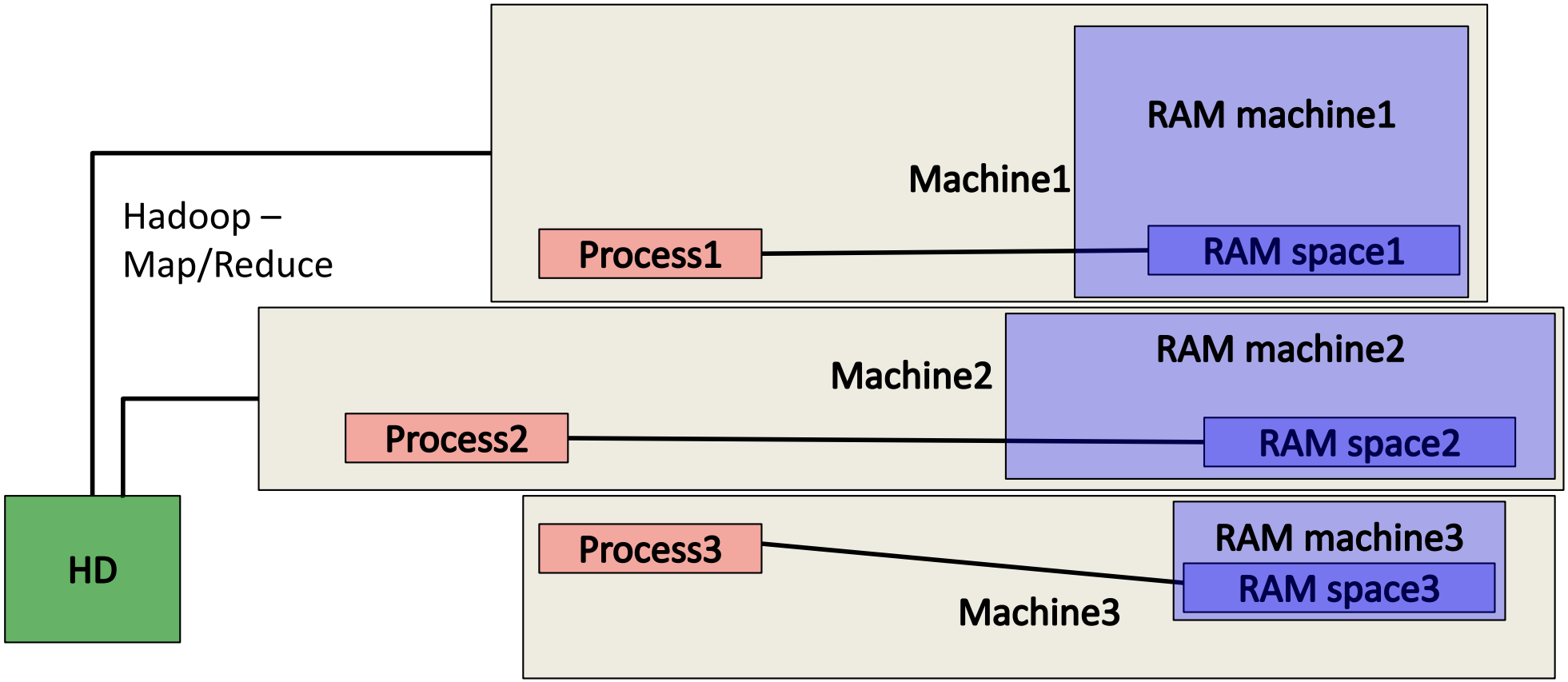
```
ssh mach1@lyon      align sample1 | count > tab1.tab
ssh mach2@nancy     align sample2 | count > tab2.tab
ssh mach3@lille     align sample3 | count > tab3.tab
```



# Parallel on a Grid (Grid5000)



# Parallel on a Grid (Grid5000)



- Distributed Memory
  - Embarrassingly Parallel

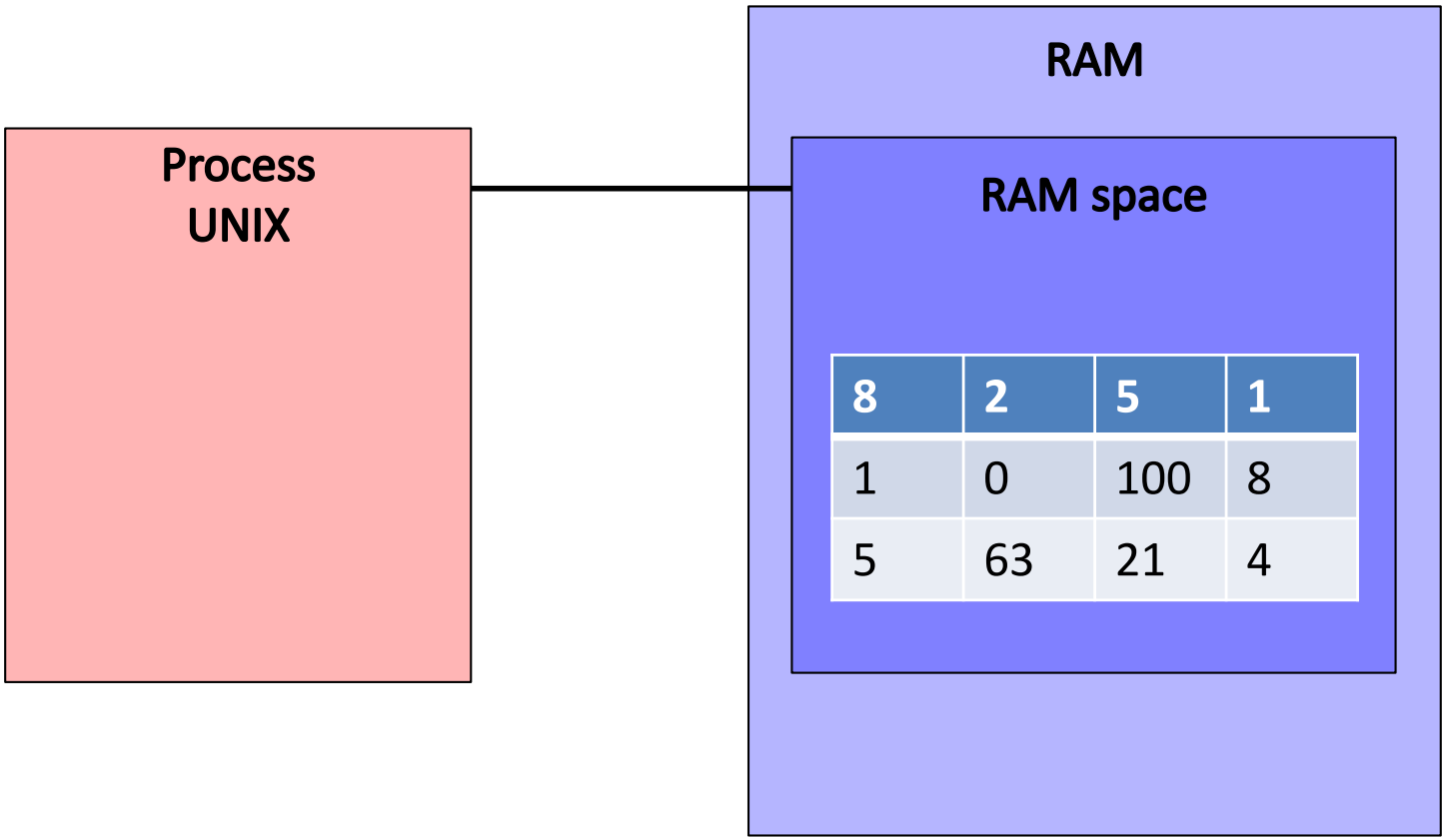
    = independent tasks

    = no communication between tasks

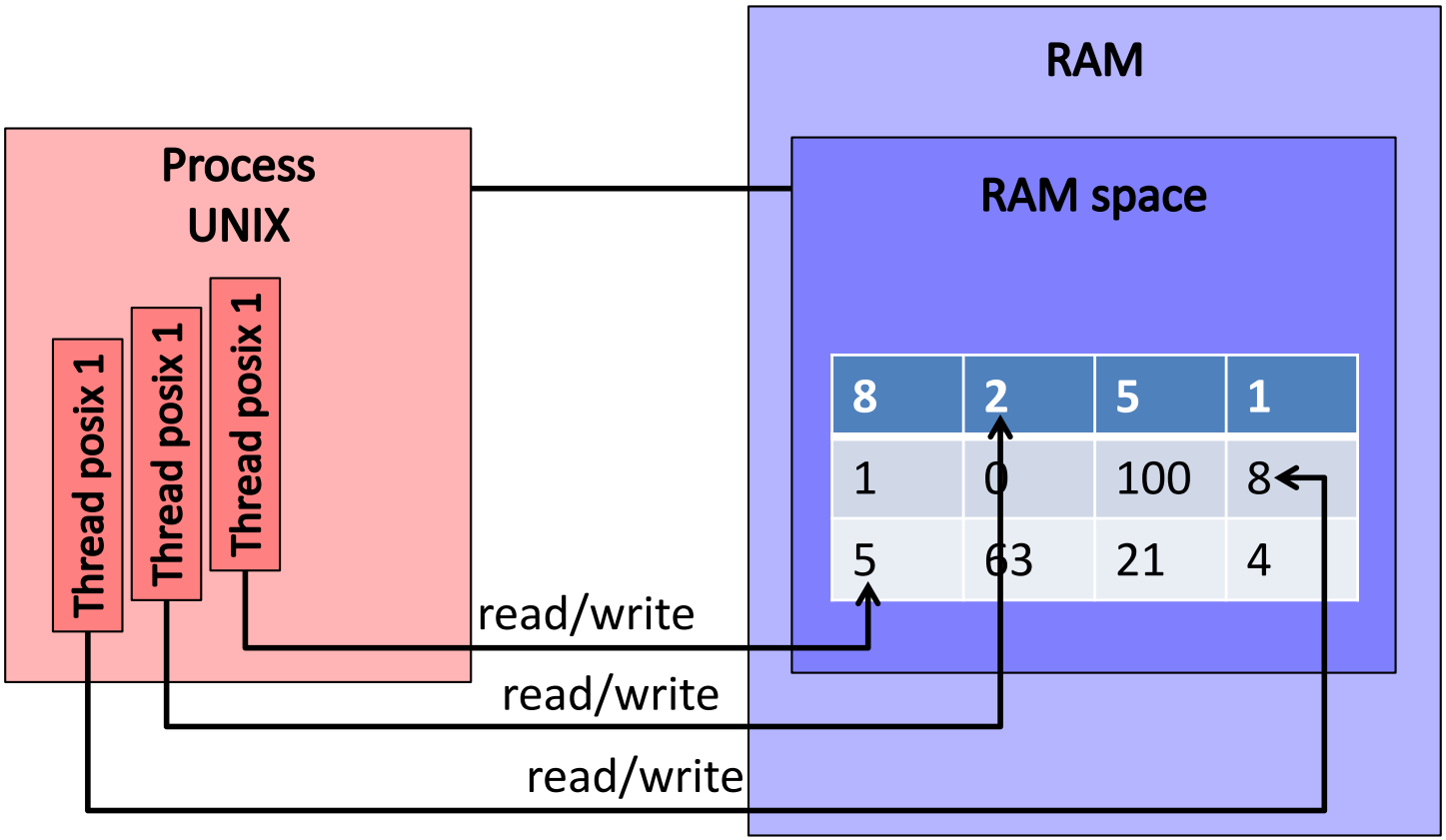
    = 99% of bioinfo/biostat tasks
  - With communications = MPI

# Shared Memory

# Shared Memory



# Shared Memory



# 2 Parallel Programming Paradigms

- Distributed Memory
  - Embarrassingly Parallel

    = independent tasks

    = no communication between tasks

    = 99% of bioinfo/biostat tasks
  - With communication = MPI
- Shared Memory
  - OpenMP
  - Posix Threads

# Vectorisation

# Vectorisation

# Summary

- Computer Architectures
  - Many cores
  - Fast access to memory
  - Slow access to hard drive
- 2 Parallel Programming Paradigms
  - Distributed memory
  - Shared memory