Goal: making pipelines scale

Scaling through parallel computing

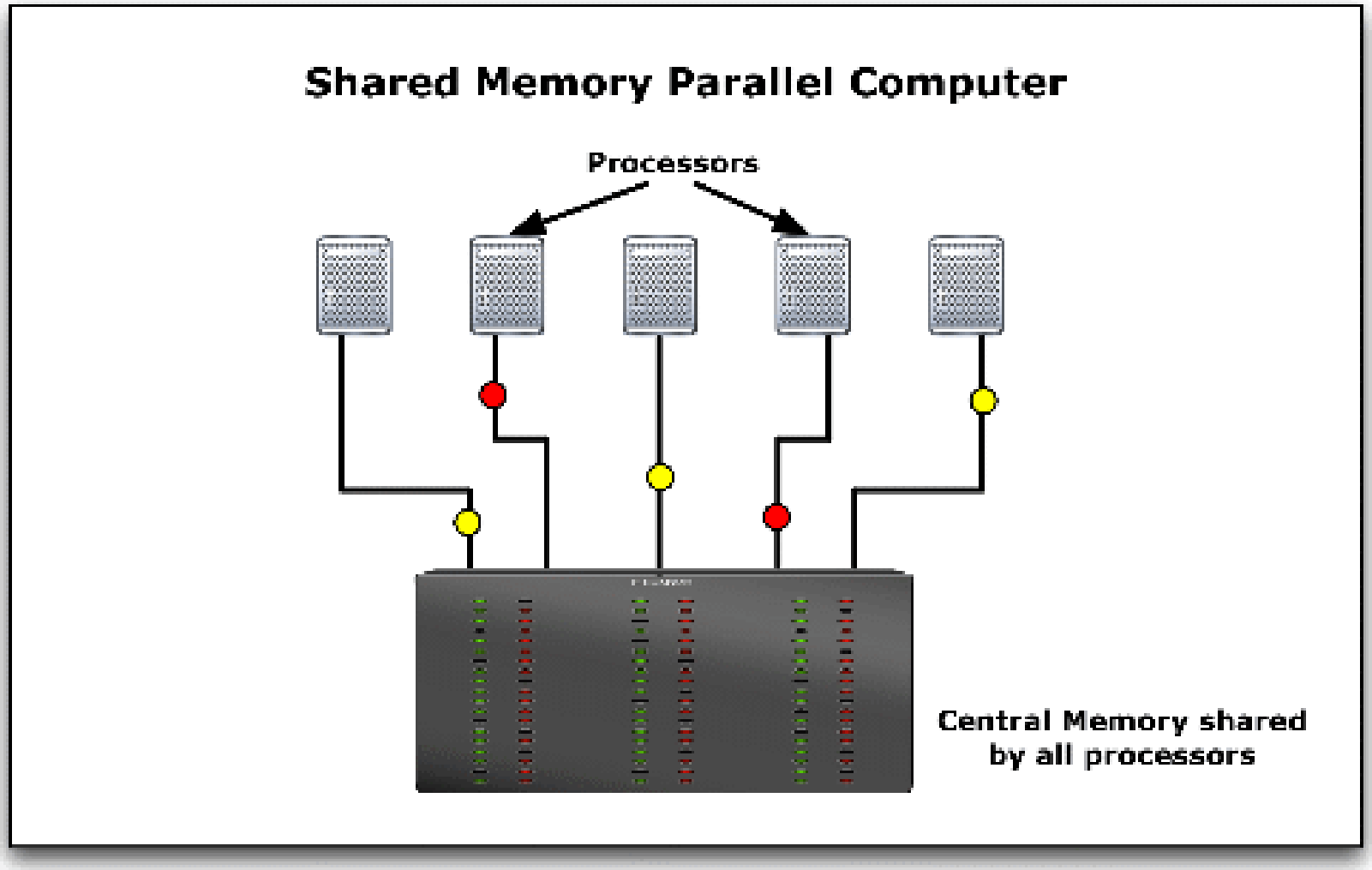
- identify: ask Sindice, finds 20 sources
- transform: download 20 RDF files in parallel
- reason: merge and answer query

Parallel computing by distribution

- identify on one machine
- transformers on second machine
- reasoner on third machine

How to do parallel computing?

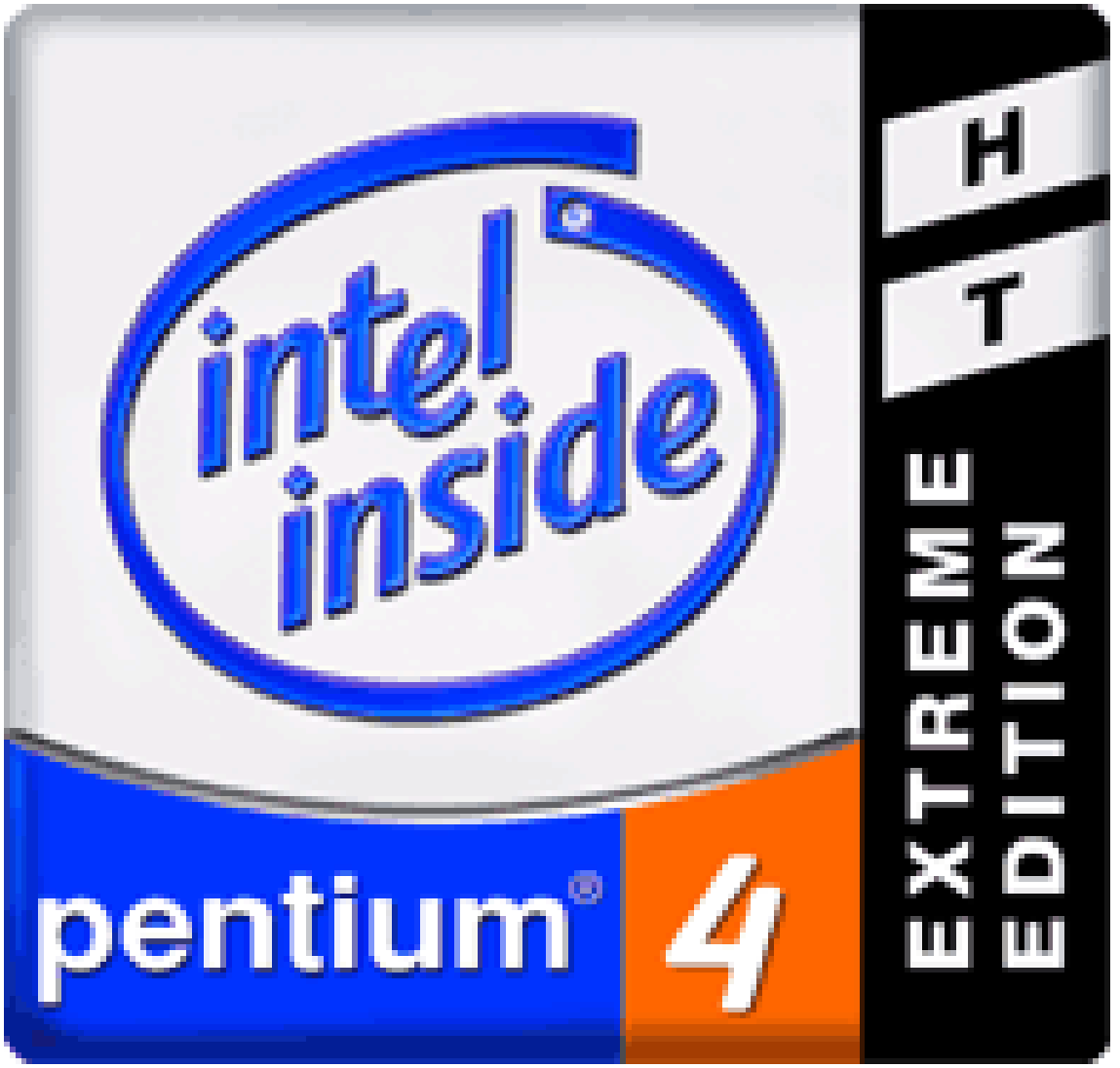
Parallel supercomputer: multiple CPUs with shared memory



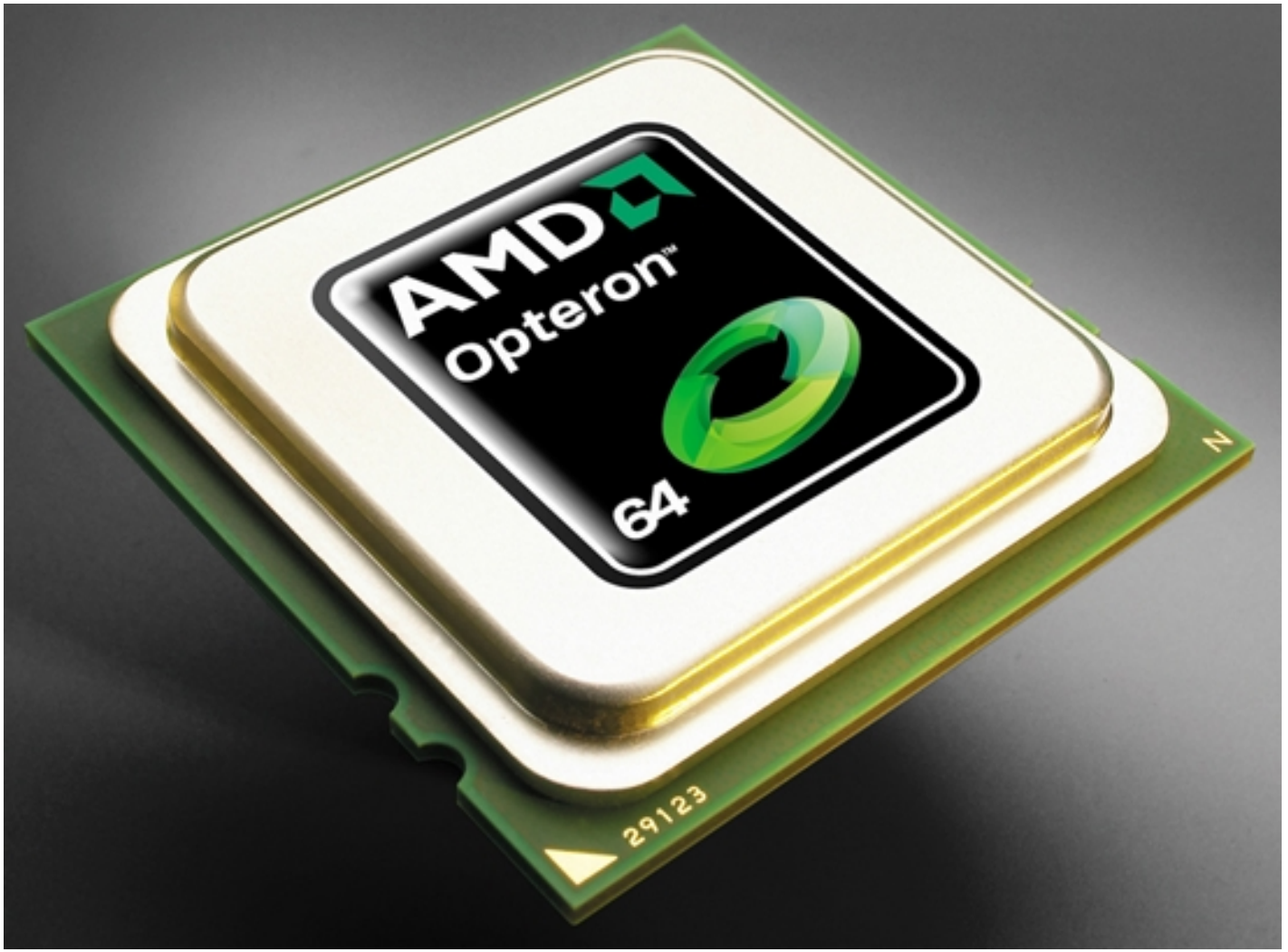
OS schedules jobs across CPUs

If you have only a laptop?

Present core as multiple CPUs



Multi-core CPU with shared memory



JVM schedules threads across cores

Thread.new()

And if you have two laptops?

Or if you have a cluster?

Or if you have a creditcard?

How will you split your program across the laptops or cluster?

#1: use a distributed language

Fibonacci

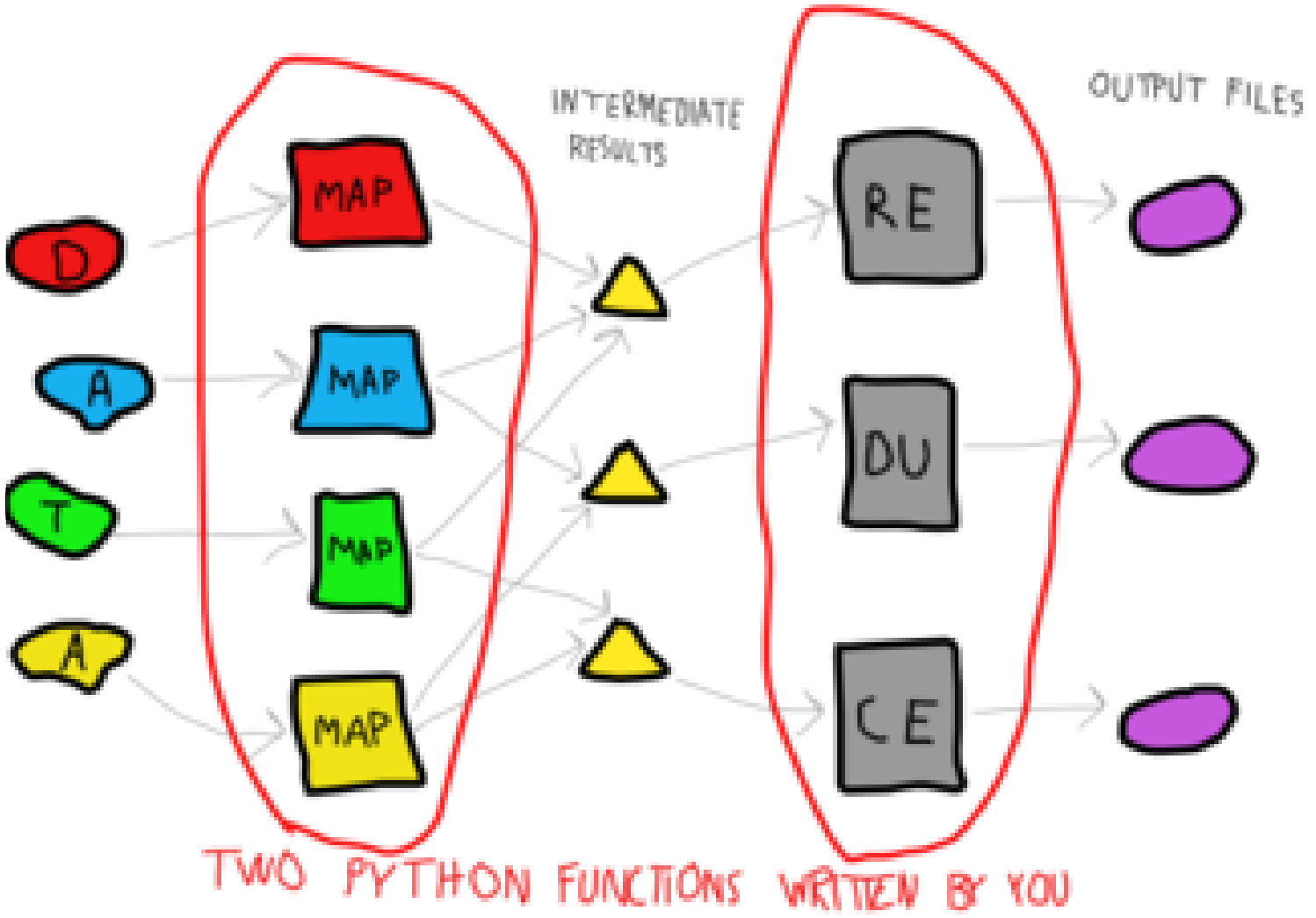
```
int fib (int n) {
if (n<2) return (n);
  else {
    int x,y;
    x = fib(n-1);
    y = fib(n-2);
    return (x+y);
  }
}
```

C elision

Cilk code

```
cilk int fib (int n) {
  if (n<2) return (n);
  else {
    int x,y;
    x = spawn fib(n-1);
    y = spawn fib(n-2);
    sync;
    return (x+y);
  }
}
```

#2: use a distributed framework



hadoop

disco

massive data - minimal code

What does LarKC offer now?

- Run plugins in parallel, on remote machines:
  - Users register computing resources
  - Decider asks remote containers for plugins
  - Platform selects machines, uploads code & input data, runs job, downloads output data

```
eu.larkc.core.pluginManager.distributed.
  ReasonerManager.runJob(
    DIGReasoner, query, statements);
```

What does LarKC offer now?

- Some plugins parallel/distributed inside:
  - Sindice identifier (threads in memory)
  - MaRVIN reasoner (peer-to-peer on cluster)
  - GATE transformer (supercomputer)
- But plugins do this on their own

- In coming year: LarKC will offer utilities to run parallel jobs on multiple machines

- plaatje scenarios
  - 2x laptops, shared memory, amazon, ...
- voorbeelden distributed programming
  - hadoop, satin, openMP
- voorbeelden deployment scenarios
  - one server, basement, cluster, EC2
- goal: how we *will* support distributed jobs
  - snippet of code for running sindice remotely

Summary

- Scaling through distribution and parallelisation

- LarKC already supports running jobs remotely
- LarKC will help you parallelise your plugin

Distributed processing

- Scaling through parallelisation
  - Parallel CPUs, shared memory
  - Thread.start
  - Eg: your notebook
- Scaling through distribution
  - Parallel CPUs, distributed memory
  - Eg: two notebooks, compute cluster, Amazon EC2
- Challenge: programming models task-dependant
  - Distributed programming is hard
  - Diverse plugins
  - Diverse deployment scenarios

What do we have now?

- Parallelisation between plugins
  - Users register computing resources
  - Decider decides to deploy plugins on remote machines
  - Platform (pipeline support system) manages data/code upload/download
    - select resources
    - prestaging
    - executing
    - poststaging
- Parallelisation inside plugins
  - Some plugins use distribution/parallelisation
  - Eg: Sindice, MaRVIN, GATE
  - No platform support (yet)
- Summary:
  - LarKC already supports running jobs remotely
  - LarKC will help you parallelise your plugin