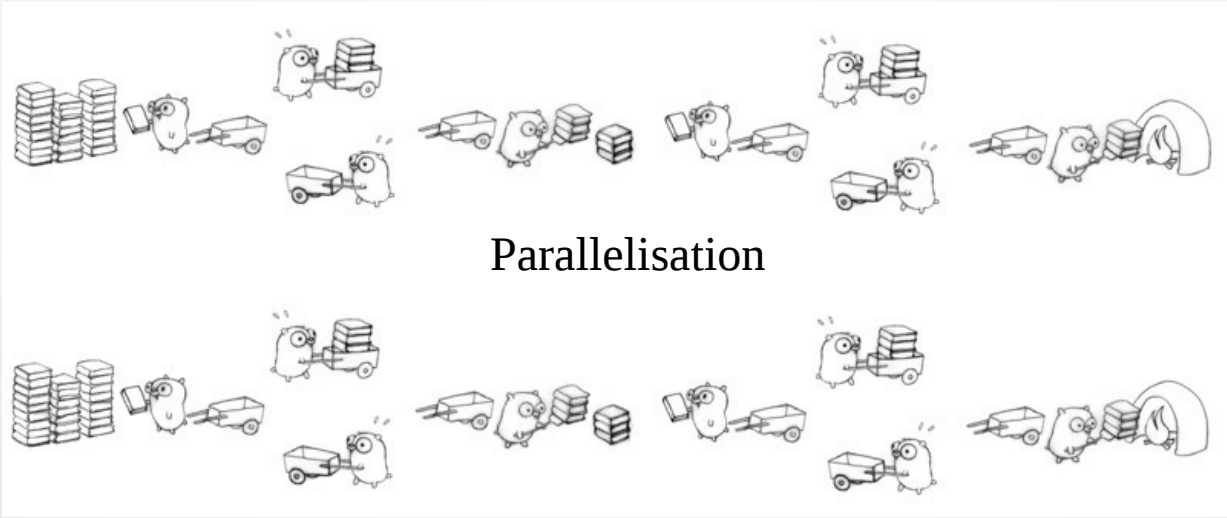
C++ Parallelisation

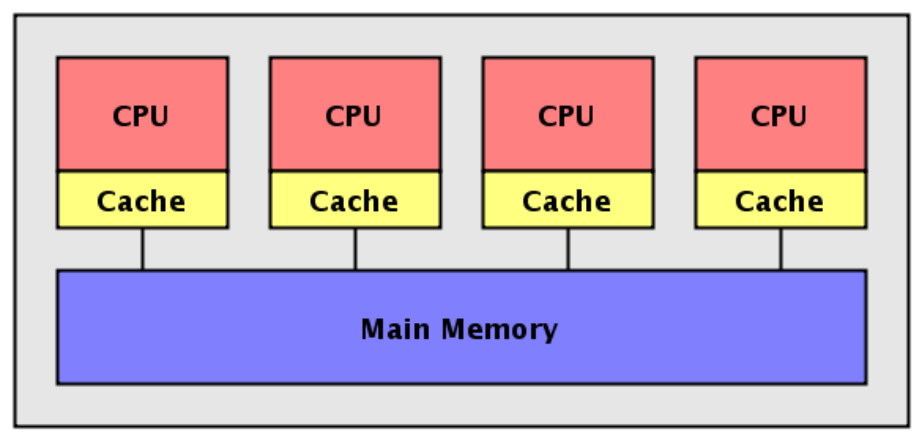
Concurrency

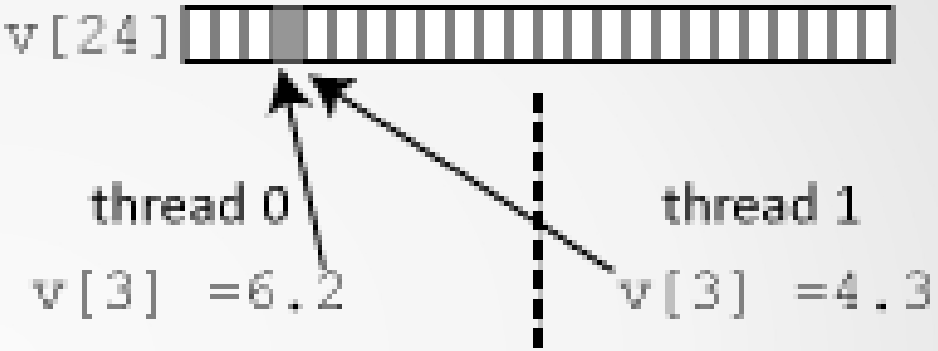


Parallelisation

Krishna Kumar

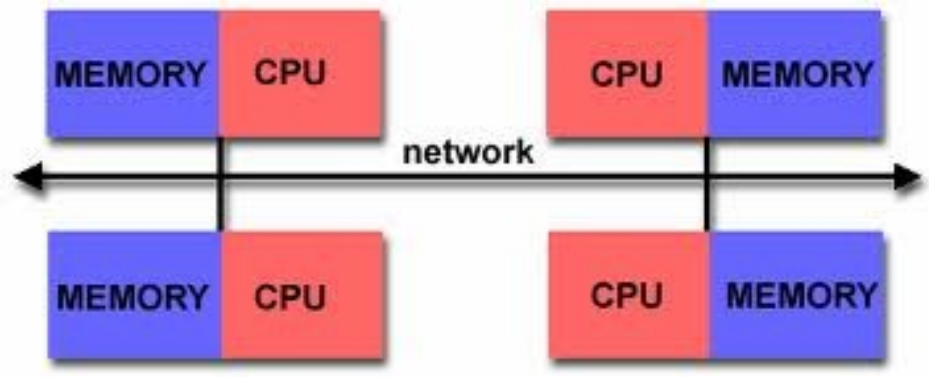
Shared memory

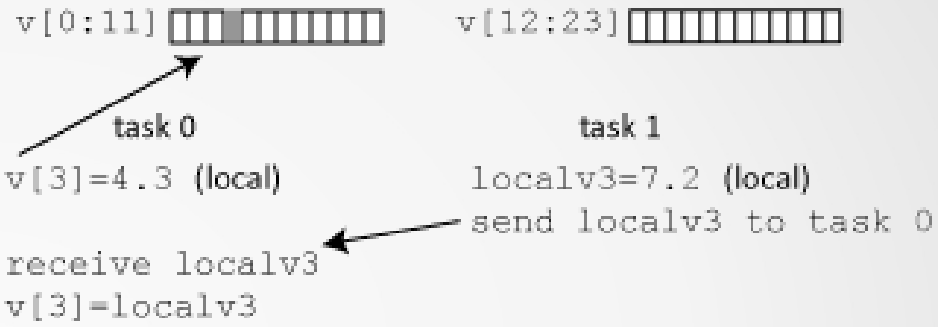




- two threads of execution can both address the same variables in a uniform manner, hereby assigning to an element of a vector whose components are in the virtual memory of the task.
- If the programmer wants thread 0 to use the value placed in the array by thread 1, he needs to use a mechanism which assures him that thread 1 has written the value before thread 0 reads it.

Distributed memory





- Each task owns part of the data, and other tasks must send a message to the owner in order to update that data.
- These may be two tasks on the same computer so that they could just share memory, but the programmer is treating them as though they were not.
- Virtual address space is not shared.

C++ libraries for parallelisation

- **Shared memory**
  - OpenMP
  - C++11 Threads
  - Posix Threads
  - Intel TBB
- **Distributed Memory**
  - MPI
- **GPU**
  - CUDA
  - OpenCL
  - OpenACC
  - AMP

- Open Multi-Processing is an API to explicitly direct multi-threaded, shared memory parallelism
- Comprised of three primary API components:
  - Compiler Directives
  - Runtime Library Routines
  - Environment Variables
- OpenMP is not:
  - For distributed memory parallel systems (by itself)
  - Guaranteed to make the most efficient use of shared memory
- **The programmer is responsible for synchronizing input and output.**