
The Abdus Salam
International Centre
for Theoretical Physics

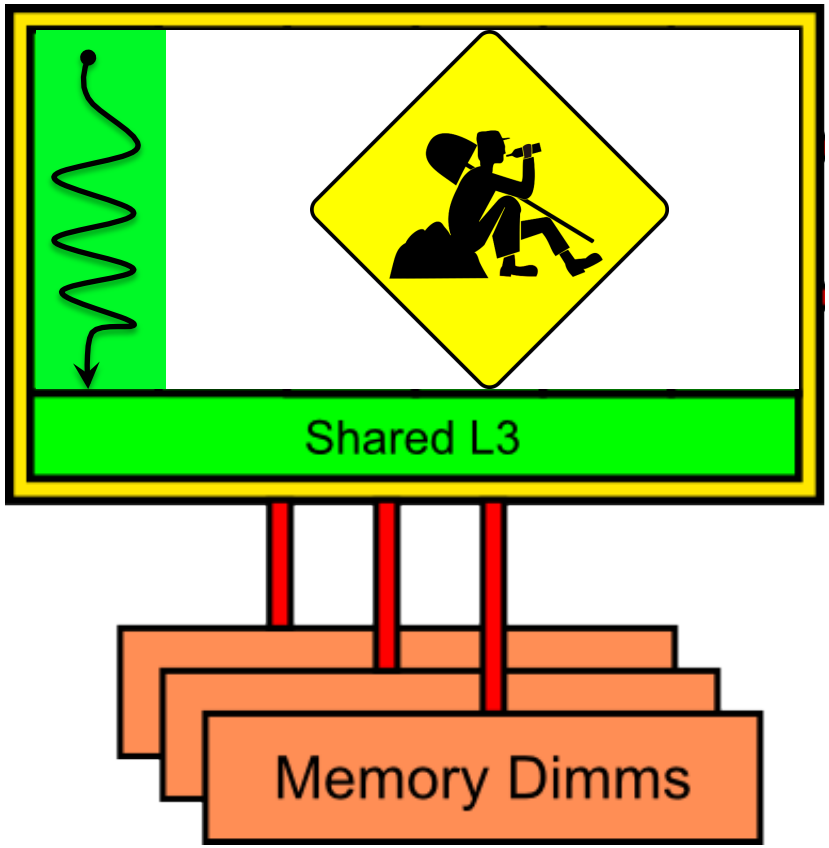

# Task Farming For Embarrassingly Parallel Processing

**Ivan Girotto – igirotto@ictp.it**

Information & Communication Technology Section (ICTS)

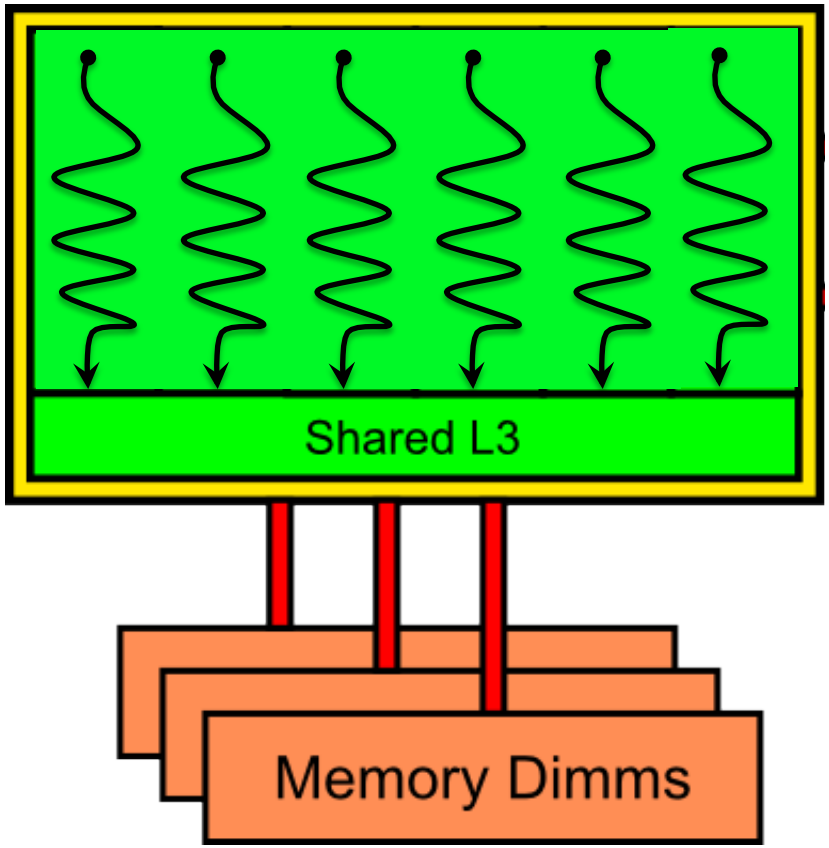
International Centre for Theoretical Physics (ICTP)

# Multi-core system Vs Serial Programming

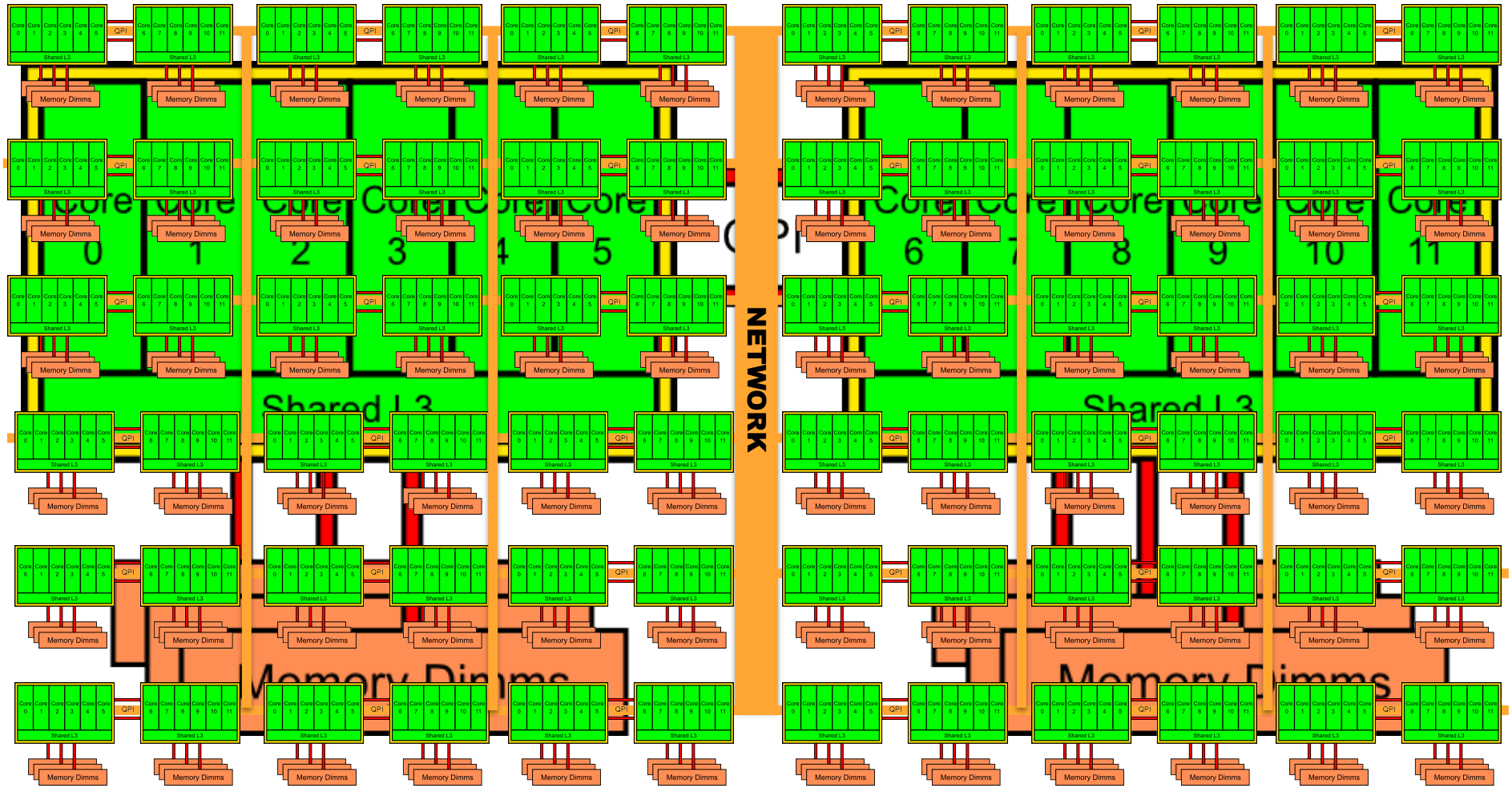



Xeon E5650 hex-core processors (12GB - RAM)

# Multi-core system Vs // Programming




Xeon E5650 hex-core processors (12GB - RAM)


NETWORK
Memory Dimms
Shared L3
QPI

# I don’t know about // Programming

Search

Search in Conferences:

Overview

Programme

Speakers

Apply here

Practical info

## Introductory School on Parallel Programming and Parallel Architecture for High-Performance Computing | (smr 2877)

**Organizers**

Shawn T. Brown, Ivan Girotto, Axel Kohlmeyer, Gavin Pringle,
ICTP Local Organizer: Ivan Girotto

The School has the goal of teaching participating scientists about modern computer hardware and programming to provide a foundation for future computational research using High-Performance Computing (HPC). Participants will go through an intensive program with focus on practical skills most relevant to users of HPC resources of all sizes. They will learn to improve the efficiency of their research codes and to parallelize them. Lectures on a selection of technical aspects of modern HPC hardware will be mixed with introductions to widely used parallel programming tools and libraries. The hands-on sessions will allow participants to practice on small example problems of general scientific interest. Example topics will cover numerical methods, parallel strategies, as well as data management.

The program is specifically addressing the needs of scientists using, writing, or modifying HPC applications will not assume, require, or provide significant IT and HPC resource management skills. It will be mainly based on fundamental HPC-relevant features in widely used scientific software for high-performance computing:

- Computer architectures for HPC and how to optimize for them
- Parallel programming tools (MPI & OpenMP)
- Portable, flexible and parallel I/O (HDF5)
- Profiling, benchmarking and debugging
- High-Performance Libraries for the Solution of Common Math Problems

**Deadline: 15/05!!**

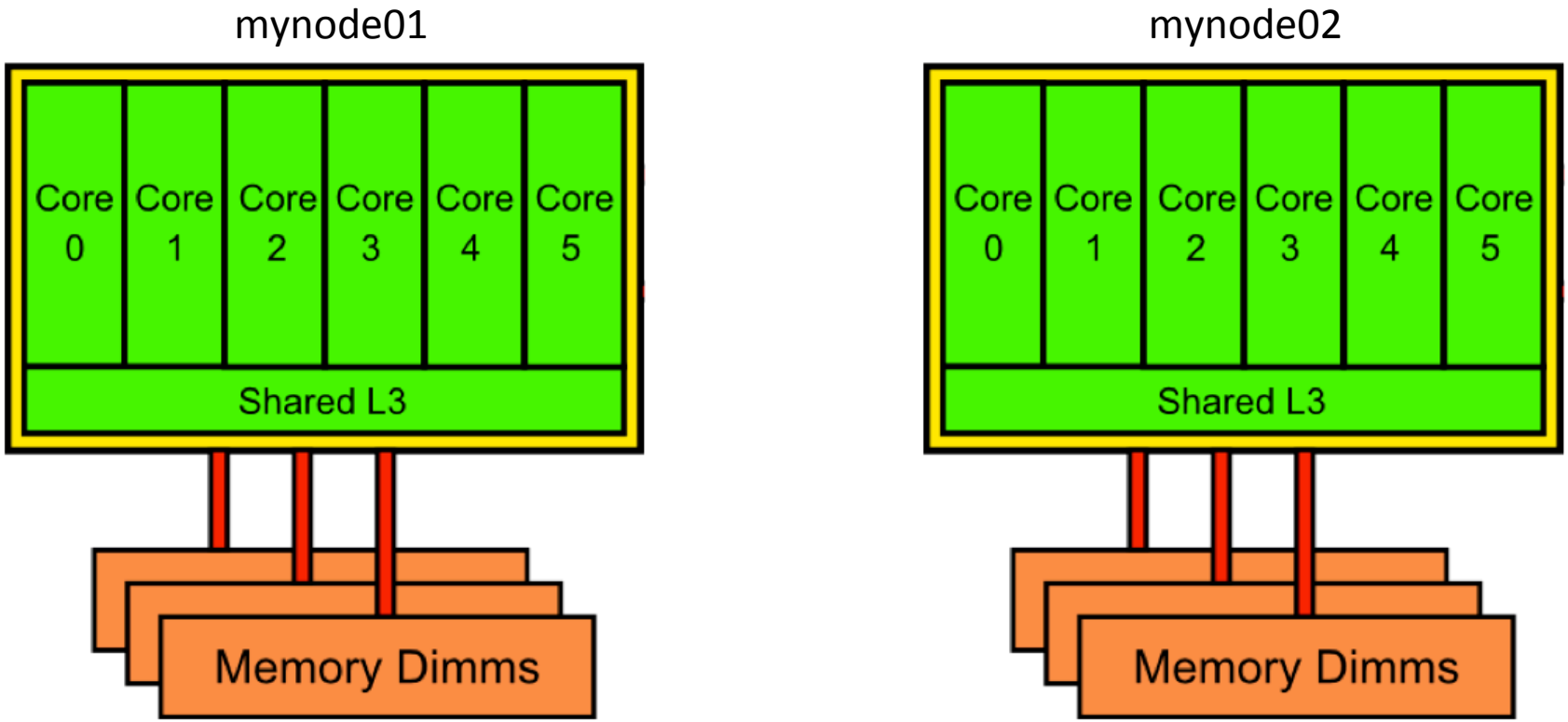
# … but I'm lucky!!

- I am working on an embarrassing parallel problem
- I can divide the work in independent tasks (no communication) that can be performed in parallel
- Quite common in Computer Graphics, Bioinformatics, Genomics, HEP, anything else requiring processing of large data-set, sampling, ensemble modeling

# Single Program on Multiple Data

- performing the same program (set of instructions) among different data
- Same model adopted by the MPI library
- A parallel tool is needed to handle the different processes working in parallel
- The MPI library provides the *mpirun* application to execute parallel instances of the same program

```
$ mpirun -np 12 my_program.x
```



```
[igirotto@mynode01 ~]$ mpirun -np 12 /bin/hostname
mynode01
mynode02
mynode01
mynode02
mynode01
mynode02
mynode01
mynode02
mynode01
mynode02
mynode01
mynode02
```

**PATH name common to all processes !!**

# Parallel Operations in Practice

- Parallel reading and computing in parallel is always allowed
- Parallel writing is extremely dangerous!
- To control the parallel flow each process should be unique and identifiable (ID)
- The OpenMPI implementation of the MPI library provides a series of environment variables defined for each MPI process

**OMPI_COMM_WORLD_SIZE** - the number of processes in this process' MPI Comm_World

**OMPI_COMM_WORLD_RANK** - the MPI rank of this process

**OMPI_COMM_WORLD_LOCAL_RANK** - the relative rank of this process on this node within its job. For example, if four processes in a job share a node, they will each be given a local rank ranging from 0 to 3.

**OMPI_UNIVERSE_SIZE** - the number of process slots allocated to this job. Note that this may be different than the number of processes in the job.

**OMPI_COMM_WORLD_LOCAL_SIZE** - the number of ranks from this job that are running on this node.

**OMPI_COMM_WORLD_NODE_RANK** - the relative rank of this process on this node looking across ALL jobs.

http://www.open-mpi.org

# In Python

```
import os
myid = os.environ['OMPI_COMM_WORLD_RANK']
[...]
```

# In BASH

```
#!/bin/bash
myid=${OMPI_COMM_WORLD_RANK}
[...]
```

```
[igirotto@mynode01 ~]$ mpirun ./myprogram.[py/sh...]
```

# Possible Applications

- Executing multiple instances on the same program with different inputs/initial cond.
- Reading large binary files by splitting the workload among processes
- Searching elements on large data-sets
- Other parallel execution of embarrassingly parallel problem (no communication among tasks)

# Conclusions

- Task Farming is a simple model to parallelize simple problems that can be divided in independent task
- The *mpirun* application aids to easily perform multiple processes, includes environment setting
- Load balancing remains a main problem, but moving from serial to parallel processing can substantially speed-up time of simulation