Trends in HPC Architectures and Parallel Programming

Giovanni Erbacci - g.erbacci@cineca.it

Supercomputing, Applications & Innovation Department - CINECA
Agenda

- Computational Sciences
- Trends in Parallel Architectures
- Trends in Parallel Programming
- PRACE
Computational Sciences

Computational science (with theory and experimentation), is the “third pillar” of scientific inquiry, enabling researchers to build and test models of complex phenomena.

Quick evolution of innovation:

- Instantaneous communication
- Geographically distributed work
- Increased productivity
- More data everywhere
- Increasing problem complexity
- Innovation happens worldwide
Technology Evolution

More data everywhere:
Radar, satellites, CAT scans, weather models, the human genome.
The size and resolution of the problems scientists address today are limited only by the size of the data they can reasonably work with.
There is a constantly increasing demand for faster processing on bigger data.

Increasing problem complexity:
Partly driven by the ability to handle bigger data, but also by the requirements and opportunities brought by new technologies. For example, new kinds of medical scans create new computational challenges.

HPC Evolution
As technology allows scientists to handle bigger datasets and faster computations, they push to solve harder problems.
In turn, the new class of problems drives the next cycle of technology innovation.
Computational Sciences today

Multidisciplinary problems

Coupled applications

- Full simulation of engineering systems
- Full simulation of biological systems
- Astrophysics
- Materials science
- Bio-informatics, proteomics, pharmaco-genetics
- Scientifically accurate 3D functional models of the human body
- Biodiversity and biocomplexity
- Climate and Atmospheric Research
- Energy
- Digital libraries for science and engineering

Large amount of data

Complex mathematical models
HPC Architectures

Performance Comes from:

Device Technology
- Logic switching speed and device density
- Memory capacity and access time
- Communications bandwidth and latency

Computer Architecture
- Instruction issue rate
  - Execution pipelining
  - Reservation stations
  - Branch prediction
  - Cache management
- Parallelism
  - Number of operations per cycle per processor
    - Instruction level parallelism (ILP)
    - Vector processing
  - Number of processors per node
  - Number of nodes in a system

G. Erbacci
HPC Architectures: Parameters affecting Performance

Peak floating point performance
Main memory capacity
Bi-section bandwidth
I/O bandwidth
Secondary storage capacity
Organization
  – Class of system
  – # nodes
  – # processors per node
  – Accelerators
  – Network topology
Control strategy
  – MIMD
  – Vector, PVP
  – SIMD
  – SPMD
HPC systems evolution

- Vector Processors
  - Cray-1
- SIMD, Array Processors
  - Goodyear MPP, MasPar 1 & 2, TMC CM-2
- Parallel Vector Processors (PVP)
  - Cray XMP, YMP, C90 NEC Earth Simulator, SX-6
- Massively Parallel Processors (MPP)
  - Cray T3D, T3E, TMC CM-5, Blue Gene/L
- Commodity Clusters
  - Beowulf-class PC/Linux clusters
  - Constellations
- Distributed Shared Memory (DSM)
  - SGI Origin
  - HP Superdome
- Hybrid HPC Systems
  - Roadrunner
  - Chinese Tianhe-1A system
  - GPGPU systems
HPC systems evolution in CINECA

1969: CDC 6600 1st system for scientific computing
1975: CDC 7600 1st supercomputer
1985: Cray X-MP / 4 8 1st vector supercomputer
1989: Cray Y-MP / 4 64
1993: Cray C-90 / 2 128
1994: Cray T3D 64 1st parallel supercomputer
1995: Cray T3D 128
1998: Cray T3E 256 1st MPP supercomputer
2002: IBM SP4 512 1 Teraflops
2005: IBM SP5 512
2006: IBM BCX 10 Teraflops
2009: IBM SP6 100 Teraflops
2012: IBM BG/Q 2 Petaflops
BG/Q in CINECA

- 10 BGQ Frame
- 10240 nodes
- 1 PowerA2 processor per node
- 16 core per processor
- 163840 cores
- 1GByte / core
- 2PByte of scratch space
- 2PFlop/s peak performance
- 1MWatt liquid cooled

-16 core chip @ 1.6 GHz
- a crossbar switch links the cores and L2 cache memory together.
- 5D torus interconnect

The Power A2 core has a 64-bit instruction set (unlike the prior 32-bit PowerPC chips used in BG/L and BG/P). The A2 core has four threads and has in-order dispatch, execution, and completion instead of out-of-order execution common in many RISC processor designs. The A2 core has 16KB of L1 data cache and another 16KB of L1 instruction cache.

Each core also includes a quad-pumped double-precision floating point unit: Each FPU on each core has four pipelines, which can be used to execute scalar floating point instructions, four-wide SIMD instructions, or two-wide complex arithmetic SIMD instructions.
Top 500: some facts

1976  Cray 1 installed at Los Alamos: peak performance 160 MegaFlop/s (10^6 flop/s)

1993  (1° Edition Top 500) N. 1  59.7 GFlop/s (10^{12} flop/s)

1997  Teraflop/s barrier (10^{12} flop/s)

2008  Petaflop/s (10^{15} flop/s): Roadrunner (LANL) Rmax 1026 Gflop/s, Rpeak 1375 Gflop/s hybrid system: 6562 processors dual-core AMD Opteron accelerated with 12240 IBM Cell processors (98 TByte di RAM)

2011  11.2 Petaflop/s: K computer (SPARC64 VIIIfx 2.0GHz, Tofu interconnect) RIKEN Japan

- 62% of the systems on the top500 use processors with six or more cores
- 39 systems use GPUs as accelerators (35 NVIDIA, 2 Cell, 2 ATI Radeon)
Moore’s law is holding, in the number of transistors
- Transistors on an ASIC still doubling every 18 months at constant cost
- 15 years of exponential clock rate growth has ended

Moore’s Law reinterpreted
- Performance improvements are now coming from the increase in the number of cores on a processor (ASIC)
- #cores per chip doubles every 18 months instead of clock
- 64-512 threads per node will become visible soon
- Million-way parallelism

Heterogeneity: Accelerators
- GPGPU
- MIC
Multi-core

Motivation for Multi-Core
- Exploits improved feature-size and density
- Increases functional units per chip (spatial efficiency)
- Limits energy consumption per operation
- Constrains growth in processor complexity

Challenges resulting from multi-core
- Relies on effective exploitation of multiple-thread parallelism
  • Need for parallel computing model and parallel programming model
- Aggravates memory wall
  • Memory bandwidth
    - Way to get data out of memory banks
    - Way to get data into multi-core processor array
  • Memory latency
  • Fragments (shared) L3 cache
- Pins become strangle point
  • Rate of pin growth projected to slow and flatten
  • Rate of bandwidth per pin (pair) projected to grow slowly
- Requires mechanisms for efficient inter-processor coordination
  • Synchronization
  • Mutual exclusion
  • Context switching
Heterogeneous Multicore Architecture

Combines different types of processors
- Each optimized for a different operational modality
  - Performance > Nx better than other N processor types
- Synthesis favors superior performance
  - For complex computation exhibiting distinct modalities

Conventional co-processors
- Graphical processing units (GPU)
- Network controllers (NIC)
- Efforts underway to apply existing special purpose components to general applications

Purpose-designed accelerators
- Integrated to significantly speedup some critical aspect of one or more important classes of computation
- IBM Cell architecture
- ClearSpeed SIMD attached array processor
Real HPC Crisis is with Software

A supercomputer application and software are usually much more long-lived than a hardware
- Hardware life typically four-five years at most.
- Fortran and C are still the main programming models
Programming is stuck
- Arguably hasn’t changed so much since the 70’s
Software is a major cost component of modern technologies.
- The tradition in HPC system procurement is to assume that the software is free.
It’s time for a change
- Complexity is rising dramatically
- Challenges for the applications on Petaflop systems
- Improvement of existing codes will become complex and partly impossible
- The use of O(100K) cores implies dramatic optimization effort
- New paradigm as the support of a hundred threads in one node implies new parallelization strategies
- Implementation of new parallel programming methods in existing large applications has not always a promising perspective
There is the need for new community codes
### Roadmap to Exascale (architectural trends)

<table>
<thead>
<tr>
<th>Systems</th>
<th>2009</th>
<th>2011</th>
<th>2015</th>
<th>2018</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Peak Flops/s</td>
<td>2 Peta</td>
<td>20 Peta</td>
<td>100-200 Peta</td>
<td>1 Exa</td>
</tr>
<tr>
<td>System Memory</td>
<td>0.3 PB</td>
<td>1 PB</td>
<td>5 PB</td>
<td>10 PB</td>
</tr>
<tr>
<td>Node Performance</td>
<td>125 GF</td>
<td>200 GF</td>
<td>400 GF</td>
<td>1-10 TF</td>
</tr>
<tr>
<td>Node Memory BW</td>
<td>25 GB/s</td>
<td>40 GB/s</td>
<td>100 GB/s</td>
<td>200-400 GB/s</td>
</tr>
<tr>
<td>Node Concurrency</td>
<td>12</td>
<td>32</td>
<td>0(100)</td>
<td>0(1000)</td>
</tr>
<tr>
<td>Interconnect BW</td>
<td>1.5 GB/s</td>
<td>10 GB/s</td>
<td>25 GB/s</td>
<td>50 GB/s</td>
</tr>
<tr>
<td>System Size (Nodes)</td>
<td>18,700</td>
<td>100,000</td>
<td>500,000</td>
<td>0(Million)</td>
</tr>
<tr>
<td>Total Concurrency</td>
<td>225,000</td>
<td>3 Million</td>
<td>50 Million</td>
<td>0(Billion)</td>
</tr>
<tr>
<td>Storage</td>
<td>15 PB</td>
<td>30 PB</td>
<td>150 PB</td>
<td>300 PB</td>
</tr>
<tr>
<td>I/O</td>
<td>0.2 TB/s</td>
<td>2 TB/s</td>
<td>10 TB/s</td>
<td>20 TB/s</td>
</tr>
<tr>
<td>MTTI</td>
<td>Days</td>
<td>Days</td>
<td>Days</td>
<td>0(1 Day)</td>
</tr>
<tr>
<td>Power</td>
<td>6 MW</td>
<td>~10 MW</td>
<td>~10 MW</td>
<td>~20 MW</td>
</tr>
</tbody>
</table>
What about parallel App?

In a massively parallel context, an upper limit for the scalability of parallel applications is determined by the fraction of the overall execution time spent in non-scalable operations (Amdahl's law).

$\text{maximum speedup tends to } \frac{1}{1 - P}$

$P = \text{parallel fraction}$

$10,000,000 \text{ core}$

$P = 0.999999$

$\text{serial fraction} = 0.000001$
Programming Models

- Message Passing (MPI)
- Shared Memory (OpenMP)
- Partitioned Global Address Space Programming (PGAS) Languages
  - UPC, Coarray Fortran, Titanium
- Next Generation Programming Languages and Models
  - Chapel, X10, Fortress
- Languages and Paradigm for Hardware Accelerators
  - CUDA, OpenCL
- Hybrid: MPI + OpenMP + CUDA/OpenCL
Trends

Scalar Application

MPP System, Message Passing: MPI

Multi core nodes: OpenMP

Accelerator (GPGPU, FPGA): Cuda, OpenCL

Hybrid codes
Message Passing domain decomposition
Ghost Cells - Data exchange

sub-domain boundaries

Ghost Cells exchanged between processors at every update

Processor 1

Processor 2

Ghost Cells

Processor 2

Processor 1
Message Passing: MPI

Main Characteristic
- Library
- Coarse grain
- Inter node parallelization (few real alternative)
- Domain partition
- Distributed Memory
- Almost all HPC parallel App

Open Issue
- Latency
- OS jitter
- Scalability
Shared memory
Shared Memory: OpenMP

Main Characteristic
• Compiler directives
• Medium grain
• Intra node parallelization (pthreads)
• Loop or iteration partition
• Shared memory
• Many HPC App

Open Issue
• Thread creation overhead
• Memory/core affinity
• Interface with MPI
OpenMP

!$omp parallel do
do i = 1 , nsl
   call 1DFFT along z ( f [ offset( threadid ) ] )
end do
!$omp end parallel do
call fw_scatter ( . . . )

!$omp parallel
do i = 1 , nzl
!$omp parallel do
   do j = 1 , Nx
      call 1DFFT along y ( f [ offset( threadid ) ] )
   end do
!$omp parallel do
   do j = 1, Ny
      call 1DFFT along x ( f [ offset( threadid ) ] )
   end do
end do
!$omp end parallel
Accelerator/GPGPU

Sum of 1D array
CUDA - OpenCL

**Main Characteristic**
- Ad-hoc compiler
- Fine grain
- offload parallelization (GPU)
- Single iteration parallelization
- Ad-hoc memory
- Few HPC App

**Open Issue**
- Memory copy
- Standard
- Tools
- Integration with other languages
Hybrid (MPI+OpenMP+CUDA+...)

Take the positive off all models
Exploit memory hierarchy
Many HPC applications are adopting this model
Mainly due to developer inertia
Hard to rewrite million of source lines

...+python)
Hybrid parallel programming

Python: Ensemble simulations

MPI: Domain partition

OpenMP: External loop partition

CUDA: assign inner loops Iteration to GPU threads

Quantum ESPRESSO

http://www.qe-forge.org/
PRACE

Partnership for Advanced Computing in Europe

http://www.prace-project.eu/

PRACE is part of the ESFRI roadmap and has the aim of creating a European Research Infrastructure providing world class systems and services and coordinating their use throughout Europe.

It covers both hardware at the multi petaflop/s level and also very demanding software (parallel applications) to exploit these systems.
PRACE Tier 0 Systems

- JUGENE
- CURIE
- HERMIT
- SuperMUC
- FERMI
- Marenostrum
PRACE: Objectives

Creation and manage a persistent, sustainable pan-European HPC service

Deployment of three to five Pflop/s systems at different European sites:

Hosting Sites: Germany, France, Italy, Spain

First Pflop/s system installed at Juelich in Germany

Establish a legal and organisational structure involving HPC centres, national funding agencies, and scientific user communities

Develop funding and usage models and establish a peer review process

Provide training for European scientists and create a permanent education programme

- Late 2009: 1PF in the top 5
- 2011-13: 3 more systems in the top 10
- 2020: the exaflop in the top 5

Call for access to PRACE HPC resources OPEN till August 2010
HPC-Europa 2: Providing access to HPC resources

HPC-Europa is a consortium of seven leading HPC infrastructures and five centres of excellence aiming at the integrated provision of advanced computational services to the European research community working at the forefront of science.

The services will be delivered at a large spectrum both in terms of access to HPC systems and provision of a suitable computational environment to allow the European researchers to remain competitive with teams elsewhere in the world.

Moreover, Joint Research and Networking actions will contribute to foster a culture of cooperation to generate critical mass for computational.

http://www.hpc-europa.eu/

HP-Europa 2: 2009 – 2012
(FP7-INFRASTRUCTURES-2008-1)
HPC-Europa 2: Access

• Provision of transnational access to some of the most powerful HPC facilities in Europe

• Opportunities to collaborate with scientists working in related fields at a relevant local research institute

• HPC consultancy from experienced staff
  • Provision of a suitable computational environment

• Travel costs, subsistence expenses and accommodation