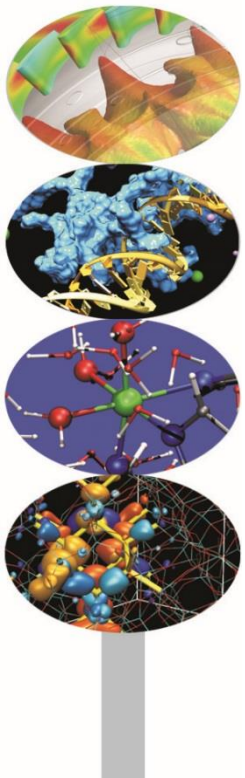
CINECA SCAI
SuperComputing Applications and Innovation

I/O: State of the art & Future developments



Giorgio Amati
SCAI

CINECA

Some questions

- Just to know each other:
  - ✓ Which is the typical I/O size you work with?
    - GB?
    - TB?
  - ✓ Is your code parallelized?
  - ✓ How many cores are you using?
  - ✓ Are you working in a small group or you need to exchange data with other researchers?
  - ✓ Which language do you use?
- Why you are here?

"Golden" rules about I/O

- Reduce I/O as much as possible: only relevant data must be stored on disks
- Save data in binary/unformatted form:
  - ✓ asks for less space comparing with ASCI/formatted ones
  - ✓ It is faster (less OS interaction)
- Save only what is necessary to save for restart or check-pointing, everything else, unless for debugging reason or quality check, should be computed on the fly.
- Dump all the quantities you need once, instead of using multiple I/O calls: if necessary use a buffer array to store all the quantities and the save the buffer using only a few I/O calls.
- Why?

What is I/O?

1. Raw data
2. **`fwritef, fscanf, fopen, fclose, WRITE, READ, OPEN, CLOSE`**
3. Call to an external library: MPI I/O, HDF5, NetCDF, ecc…
4. Scalar/parallel/network Filesystems
    1. I/O nodes and Filesystem cache
    2. I/O network (IB, SCSI, Fibre, ecc..)
    3. I/O RAID controllers and Appliance (Lustre, GPFS)
    4. Disk cache
    5. FLASH/Disk (one or more Tier)
5. Tape

Latencies

- I/O operations involves
  - ✓ OS & libraries
  - ✓ IO devices (e.g. RAID controllers)
  - ✓ Disks
- I/O latencies of disks are of the order of microseconds
- RAM latencies of the order of 100-1000 nanoseconds
- FP unit latencies are of the order of 1-10 nanoseconds

→ I/O very slow compared to RAM of FP unit

→ In the same time you could compute O(10^6) Flops

Architectural trends

Peak Performance		Moore law
FPU Performance		Dennard law
Number of FPUs		Moore + Dennard
App. Parallelism		Amdahl's law

Architectural trends

		2020 estimates
Number of cores		10^9
Memory x core		100Mbyte or less
Memory BW/core		500GByte/sec
Memory hierachy		Reg, L1, L2, L3, …

Architectural trends

		2020 estimates
Wire BW/core		1GByte/sec
Network links/node		100
Disk perf		100Mbyte/sec
Number of disks		100K

An up-to-date example/1

- CFD code
- Writing on disc, serial, formatted vs. unformatted
  - ✓ Using Intel Xeon E5-2697 (BDW), 36 Core, HT off, 2.3 GHz
  - ✓ Using Intel Phi 7250 (KNL), 68 Core, HT on, 1.4 GHz
- Both CPU share the same filesystem
- Time (lower is better)

	size	BDW	KNL	Ratio
formatted	211 MB	58″	468″	8.0
binary	1200 MB	1.20″	1.25″	-

An up-to-date example/2

- another CFD code
- Reading from disc, serial formatted
  - ✓ Using Intel Xeon E5-2697 (BDW), 36 Core, HT off, 2.3 GHz
  - ✓ Using Intel Phi 7250 (KNL), 68 Core, HT on, 1.4 GHz
- Both CPU share the same filesystem
- Time (lower is better)

	size	BDW	KNL	Ratio
formatted	3.0 GB	292”	1597”	5.5

What is parallel I/O?

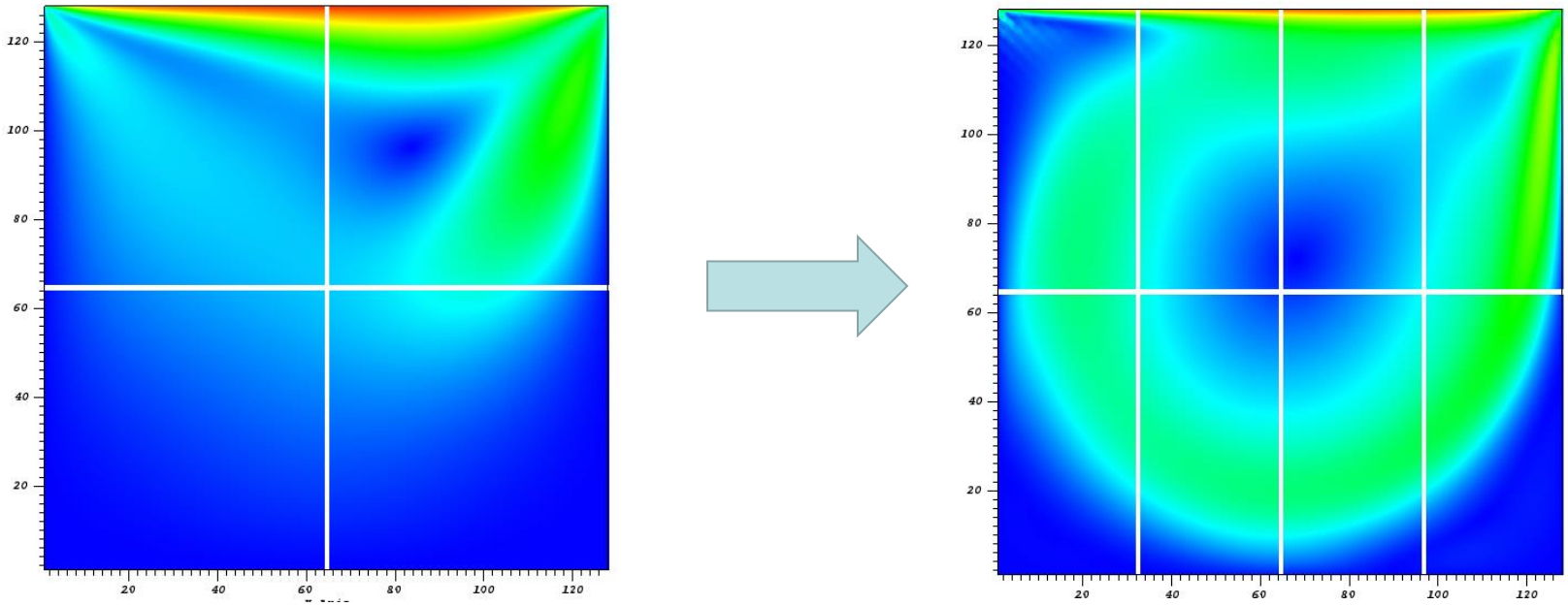
- A more correct definition in MPI-IO presentation
- Serial I/O
  - ✓ 1 task writes all the data
- Parallel I/O
  - ✓ All task write its own data in a different file
  - ✓ All task write its own data in a single file

- MPI/IO, HDF5, NetCDF, CGNS, ADIOS……

Why parallel I/O?

- New Architectures: many-many core (up to 10^9)
- As the number of task/threads increases I/O overhead start to affect performance
- I/O (serial) will be a serious bottleneck
- Parallel I/O is mandatory else no gain in using many-many core
- Other issues:
  - ✓ domain decomposition
  - ✓ data format: ASCII vs binary
  - ✓ endianess
  - ✓ Block-size
  - ✓ data management

I/O: Domain Decomposition



- Restarting a simulation using a different number of tasks
  1. Pre/Post processing
     - ✓ (merging & new decomposition → OpenFoam)
  2. Serial dump/restore
  3. Parallel I/O

I/O: ASCII vs. binary/1

- ASCII is more demanding respect binary in term of disk occupation
- Numbers are stored in bit (single precision floating point number → 32 bit)

- 1 single precision on disk (binary) → 32 bit
- 1 single precision on disk (ASCII) → 80 bit
  - 10 or more **`char`** (**`1.23456e78`**)
  - Each char asks for 8 bit

✓ Not including spaces, signs, return, …
✓ Moreover there are rounding errors: binary doesn't map with decimal

I/O: ASCII vs. binary/2

- Some figures from a real world application: **openFOAM**
- Test case: 3D Lid Cavity, 200^3, 10 dump

- Formatted output (ASCII, default)
  - ✓ Total occupation: 11 GB
- Unformatted output (binary)
  - ✓ Total occupation: 6.1 GB

- A factor 2 in disk occupation!!!!

I/O: endianess

- IEEE standard set rules for floating point operations
- But set no rule for data storage
- Single precision FP: 4 bytes (**B0**,B1,B2,B3)
  - ✓ Big endian (IBM): **B0** B1 B2 B3
  - ✓ Little endian (INTEL): B3 B2 B1 **B0**
- Solutions:
  - ✓ Hand made conversion
  - ✓ Compiler flags (intel, pgi)
  - ✓ I/O libraries (HDF5)

I/O: blocksize

- The blocksize is the basic (atomic) storage size
- One file of 100 bit will occupy 1 blocksize, that could be > 4MB

```
ls -lh TEST_1K/test_1
-rw-r--r-- 1 gamati01  10K 28 gen 11.22 TEST_1K/test_1
…
du -sh TEST_1K/test_1
512K        TEST_0K/test_1
…
du -sh TEST_1K/
501M        TEST_10K/
…
```

- Always use **`tar`** commando to save space

```
ls -lh test.tar
-rw-r--r-- 1 gamati01 11M  5 mag 13.36 test.tar
```

Example: seismic signature

- 1 signal for each day for each station
- Cross-correlation with a template
- Typical figures:
  - ✓ 1 year statistics
  - ✓ About 50 stations
  - ✓ About 1000 templates (very little in size)

…

```
gamati01@node013.pico:[AMATRICE_2015]$ du -sh template/
110M    template
gamati01@node013.pico:[AMATRICE_2015]$ ls -lh template/3.IV.mseed
-rw-r--r-- 1 gamati01 1,0K  9 nov  2016 template/3.IV.mseed
gamati01@node013.pico:[AMATRICE_2015]$ tar -cvf tutto.tar template
gamati01@node013.pico:[AMATRICE_2015]$ ls -lh  tutto.tar
-rw-r--r-- 1 gamati01 interactive 2,0M 18 mag 12:23 tutto.tar
```

…

- Disk usage → 110 MB
- Real space → 2 MB

I/O: managing data

- TB of different data sets
- Hundreds of different test cases
- Metadata
- Share data among different researchers
  - ✓ different tools (e.g. visualization tools)
  - ✓ different OS (or dialect)
  - ✓ different analysis/post processing

- You need a common "language"
  - ✓ Use I/O libraries
  - ✓ Invent (and maintain) your own data format

Some figures/1

Simple CFD program, just to give you an idea of performance loss due to I/O.

- 2D Driven Cavity simulation
- 2048*2048, Double precision (about 280 MB), 1000 timestep
- Serial I/O = 1.5”
  - ✓ 1% of total serial time
  - ✓ 16% of total time using 32 Tasks (2 nodes) → 1 dump = 160 timestep
- Parallel I/O = 0.3” (using MPI I/O)
  - ✓ 3% of total time using 32 Tasks (2 Nodes) → 1 dump = 30 timestep
- An what using 256 tasks?

Some figures/2

- Global Performance to dump huge simulation using Galileo:
- same code with threes different I/O strategies

- RAW (512 files, 2.5GB per file)
  - ✓ Write: 3.5 GB/s
  - ✓ Read: 5.5 GB/s
- HDF5 (1 file, 1.2TB)
  - ✓ Write: 2.7 GB/s
  - ✓ Read: 3.1 GB/s
- MPI-IO (19 files, 64GB per file)
  - ✓ Write: 3.1 GB/s
  - ✓ Read: 3.4 GB/s

Some strategies

- ✓ I/O is the bottleneck → avoid when possible
- ✓ I/O subsystem work with locks → simplify application
- ✓ I/O has its own parallelism → use MPI-I/O
- ✓ I/O is slow → compress (to reduce) output data
- ✓ Raw data are not portable → use library
- ✓ I/O C/Fortran APIs are synchronous → use dedicated I/O tasks

Application DATA are too large → analyze it "on the fly", (e.g. re-compute vs. write)

At the end: moving data

- Now I have hundreds of TB. What I can do?
  - Storage using Tier-0 Machine is limited in time (e.g. PRACE Project data can be stored for extra 3 Month)
  - Data analysis can be time consuming (eyen years)
  - I don’t want to delete data
  - I have enough storage somewhere else?

How can I move data?

Moving data: theory

- BW requirements to move Y Bytes in Time X

Bits per Second Requirements					
10PB	25,020.0 Gbps	3,127.5 Gbps	1,042.5 Gbps	148.9 Gbps	34.7 Gbps
1PB	2,502.0 Gbps	312.7 Gbps	104.2 Gbps	14.9 Gbps	3.5 Gbps
100TB	244.3 Gbps	30.5 Gbps	10.2 Gbps	1.5 Gbps	339.4 Mbps
10TB	24.4 Gbps	3.1 Gbps	1.0 Gbps	145.4 Mbps	33.9 Mbps
1TB	2.4 Gbps	305.4 Mbps	101.8 Mbps	14.5 Mbps	3.4 Mbps
100GB	238.6 Mbps	29.8 Mbps	9.9 Mbps	1.4 Mbps	331.4 Kbps
10GB	23.9 Mbps	3.0 Mbps	994.2 Kbps	142.0 Kbps	33.1 Kbps
1GB	2.4 Mbps	298.3 Kbps	99.4 Kbps	14.2 Kbps	3.3 Kbps
100MB	233.0 Kbps	29.1 Kbps	9.7 Kbps	1.4 Kbps	0.3 Kbps
	1H	8H	24H	7Days	30Days

moving data: some figures/1

- Moving outside CINECA
  - ✓ **`scp`** → 10 MB/s
  - ✓ **`rsync`** → 10 MB/s
- I must move 50TB of data:
  - ✓ Using **`scp`** or **`rsync`** → 60 days
- No way!!!!!
- Bandwidth depends on network you are using. Could be better, but in general is even worse!!!

moving data: some figure/2

- Moving outside CINECA
  - **`gridftp`** → 100 MB/s
  - **`globusonline`** → 100 MB/s
- I must move 50TB of data:
  - Using **`gridftp/globusonline`** → 6 days
- Could be a solution…

- Note
  - We get these figures between CINECA and a remote cluster using a 1Gb Network (> 2 years ago)

moving data: some hints

- **Size matters**: moving many little files cost more then moving few big files, even if the total storage is the same! → latency
- Moving file from Fermi to a remote cluster via Globusonline

Size	Num. of files	Mb/s
10 GB	10	227
100 MB	1000	216
1 MB	100000	61

✓ You can loose a factor 4, now you need 25 days instead of 6 to move 50TB!!!!!

moving data: some hints

- Plan your data-production carefully
- Plan your data-production carefully (again!)
- Plan your data-production carefully (again!)
- Clean your dataset from all unnecessary stuff
- Use unformatted/binary
- Compress all your ASCII files
- Use **`tar`** to pack as much data as possible
- Organize your directory structure carefully
- Synchronize with **`rsync`** in a periodic way
  - ✓ We had a user who wants to move 20TB distributed over more then 2'000'000 files…
  - ✓ **`rsync`** asks many hours (about 6) only to build the file list, without any remote synchronization at all