Computational Chemistry codes usage on IBM SP and IBM Linux Cluster

Sigismondo Boschi, CINECA

May 31 - June 3, 2005

Two typical applications

- Gaussian
- NWchem

SCF-HF example

- The complexity of a computational chemistry system is determined by the number of basis functions: for the simplest engine you do need to evaluate:
- $N^4/8$ integrals: $O(N^4)$ each of them will be used more times to build the Fock matrix.
- Apply the SCF procedure to an NxN matrix: $O(N^{2.0\text{-}3.0})$
- In general (not only for SCF) you can distinguish three approaches:

Integral evaluation approaches

- **IN CORE**: all the integrals are evaluated once and put in memory. Then the matrix is build from them;
- **DIRECT**: any time you need an integral it is evaluated, but never stored;
- **SEMI-DIRECT**: some of the integrals are stored in memory or on disk, the others are evaluated when needed;

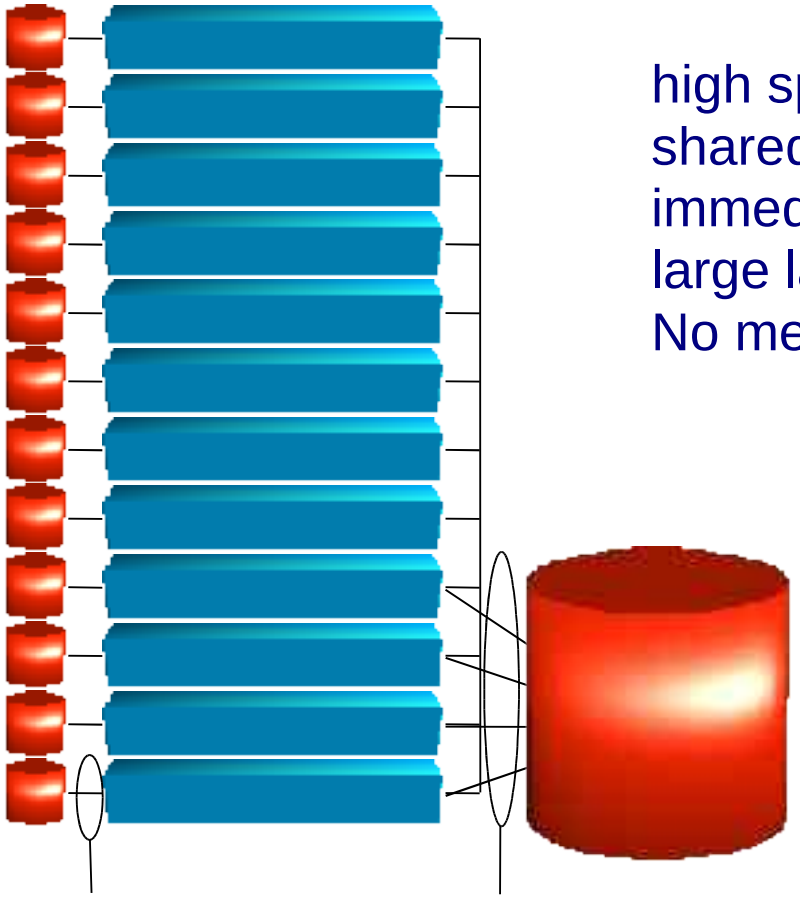
How do I choose? It depends on the characteristics of your computer. On today architectures *standard* semi-direct is the most common choice.

The “Gaussian case”

- Its story starts in 70’s
- Some “re-design” in late 80’s:
  common -> arguments
- MANY contributors to “side” engines
- 500000+ lines of code in Gaussian03
- **Computing engines based on I/O**
- 90% of runs use the SCF core: standard runs
- It has been written before the invention of parallel computing!
- Parallelised partially (SCF, optimizations…) using OpenMP and/or Linda

I/O subsystems for parallel systems

scalable
dedicated
distributed cache
difficult to export data



high speed
shared in bandwidth
immediate to use
large latency
No memory cache

30 MB/s
on CLX

200 -> 500 MB/s
on CLX

Comparing HF-SCF with NWChem

- cc-pvtz amoxycillin, 44 atoms, 449 basis functions, C1 symmetry.
- 10^9 integrals (10GBs workarea)
- Integral storage choices:
  - **local disks**: semidirect, with minimal usage of memory
  - **gpfs**: semidirect, with minimal usage of memory
  - **no disk**: direct if integrals do not fit into default memory; in-core otherwise
  - **no disk + mem**: tell NWchem to use up to 100MW per CPU
  - **local disk + mem**: use local disks and 100MW cache on every CPU.

SP HPS Power4 1.3 GHz

vs

Cluster Linux Myrinet Xeon 3.055 GHz

- Just one 32 processors (1 p690 node, no HPS usage) run:

- Standard (GPFS):

CLX: 199.7 sec SP4: 240.7 sec

- Specifing memory:

CLX: 59.1 sec SP4: 79.9 sec

What does it mean “standard”?

```
scratch_dir /scratch/abc0
memory total 900 mb

… molecule specifications …

task scf
```

means also:

global 450
heap 225
stack 225

"expert usage"

```
scratch_dir /scratch/abc0
memory total 900 global 100 mb

… molecule specifications …

scf
  RHF
  nopen 0
  semidirect memsize 100000000
end

task scf
```

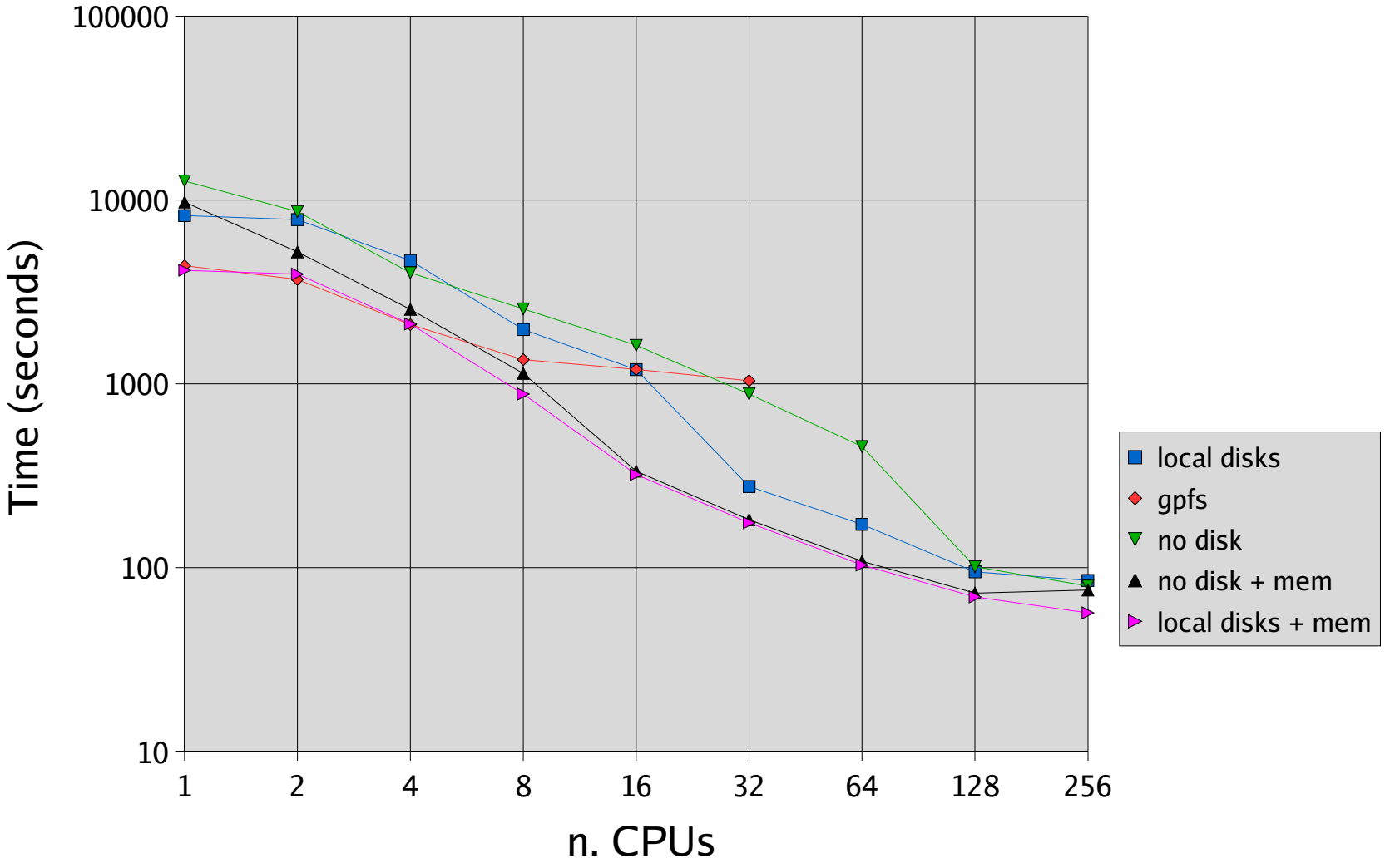
"local disk":
/scratch_local/abc0

means also:

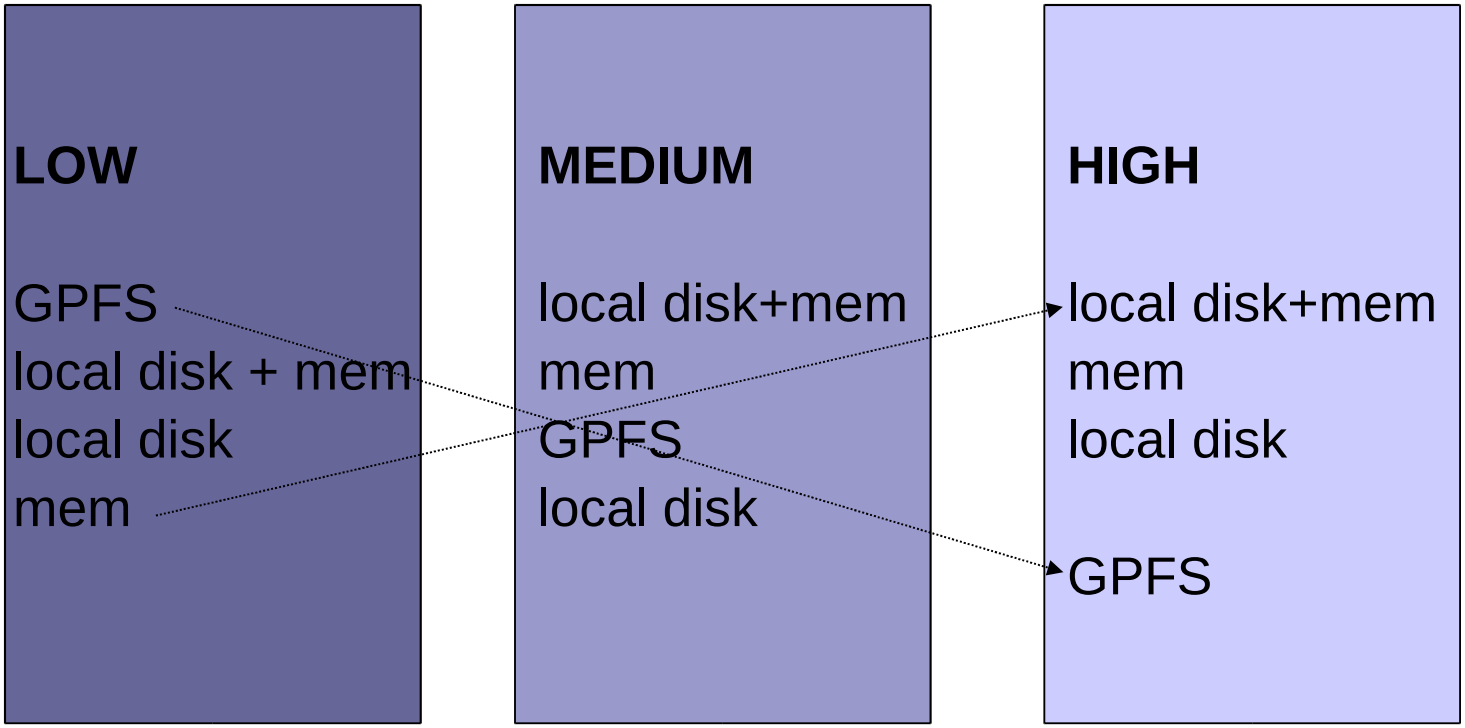
heap 400
stack 400

"no disk":
disksize 0

Timing of the same run (cc-pvdz amoxycillin, 449 basis functions, 10^9 integrals, 10GBs workarea) with different choices of integral storage



what's better?

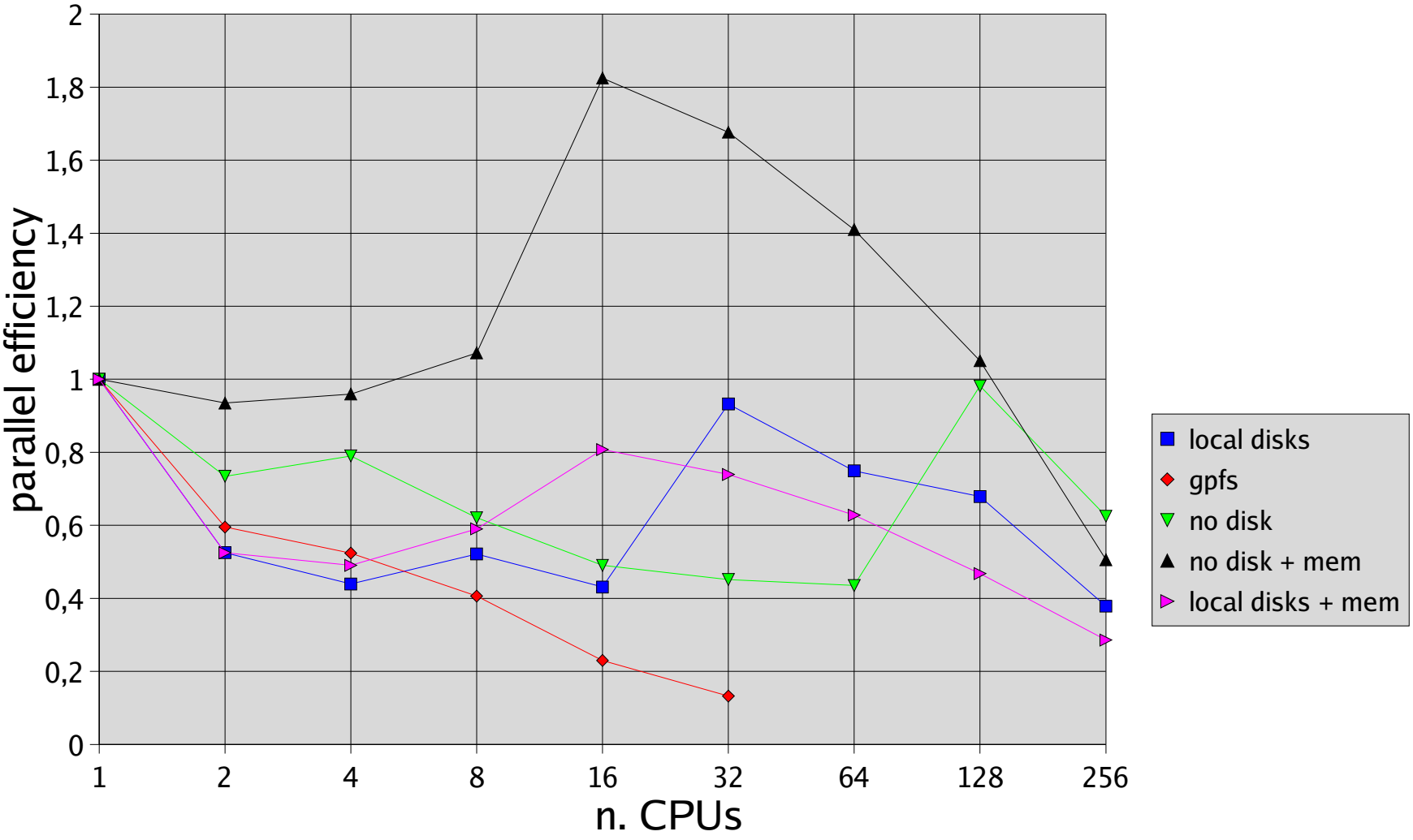


It depends on the parallelism level but it is NEVER worthwhile to recompute the integral respect to storing them except for one case...

what's the *standard* with NWchem?

- semidirect on GPFS – the worst for high scalability!!!
- Even if a modern code, the *standard* usage let you have standard capabilities… that are not compatible with high parallelism.
- even a good piece of code need an expert “driver”.
- Computational chemists needs to get involved in the problem;
- Computational chemistry codes as well;
- Dealing correctly with memory is the problem.

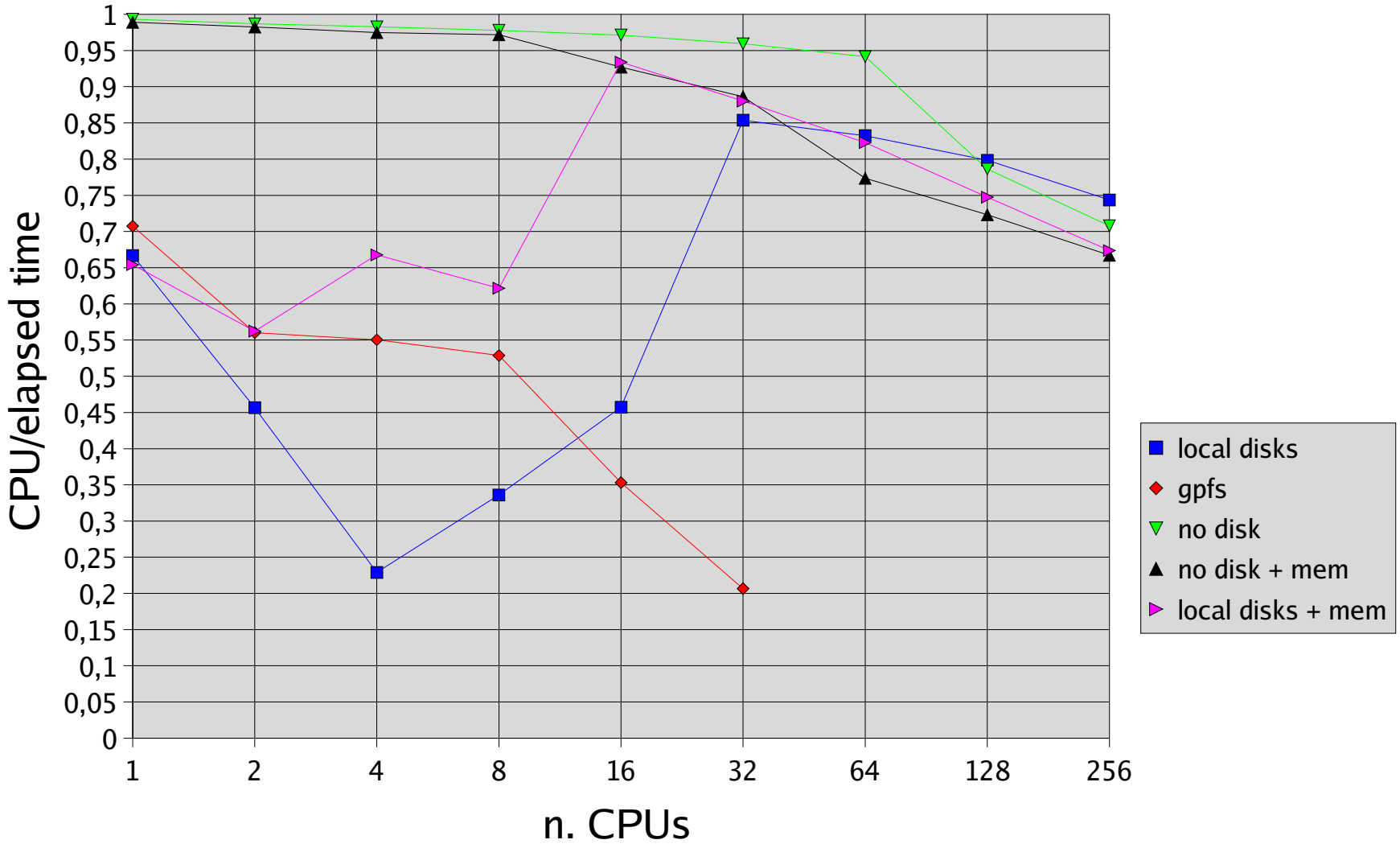
Parallel efficiency



scalability

- The methods that scale better are the ones that do not use disk at all, despite their absolute performance:
  - I/O
  - context switches
  - OS activity
  - variable task times
- The worst is GPFS! It is able of good performance but it is not scalable! Furthermore it is shared => variable performance.

cpu/elapsed time ratio



What's up with I/O?

- it is not useful
- often it is negative
- best performance for high scalability is without I/O
- I/O is a killer for high scalability: it needs context switches

an historical problem...

- Computational chemistry kernels:
  - in core = all integrals in memory, including zeros;
  - direct = integrals evaluated as needed;
  - semi-direct = AO integrals recomputed as needed, MO quantities stored temporarly on disk

ATTENTION! this means that it is assumed that:

- either you have enought memory to do the in-core stuff
- or you do not, and you need disk (semidirect) or CPU (direct).
- Memory is just a “cache” to disk…

ALIAS: you must specify by hand “do not use disk”... if you can at all, because today parallel computers...

... have really a lot of distributed, fast and *unused* memory