A quantum chemistry calculation distributed among computing facilities with Quantum Package

A. Scemama[1], P. Bousquet-Melou[2], M.-S. Cabot[2], N. Renon[3]

October 2019

[1]LCPQ: Lab. Chimie et Physique Quantiques, IRSAMC, UPS/CNRS, Toulouse
[2]CRIANN: Centre Régional Informatique et d'Applications Numériques de Normandie, Saint-Étienne-du-Rouvray
[3]CALMIP: Calculs en Midi-Pyrénées, Toulouse

Abstract

Abstract

We have shown we are able to make *routinely* distributed calculations for quantum chemistry using 20 nodes of CRIANN and 20 nodes CALMIP.

Outline

- Presentation of the method
  - Selection
  - Davidson's iterations
- Parallelism / Experimental setup
- Results
  - Selection
  - Davidson's iterations

Scientific problem: Schrödinger's equation

$$\mathbf{H}\,\mathbf{\Psi} = E\,\mathbf{\Psi} \quad ; \quad \mathbf{\Psi} = \sum_{i=1}^{N_{\text{det}}} \Psi_i \mathbf{D}_i$$

Find the best possible approximation of $\mathbf{\Psi}$and E

- Find approximate lowest eigenpair(s) and of the Hamiltonian matrix $\mathbf{H}$
- N can be as large as 10^{30} for small molecules (exponential scaling)
- Eigenvector(s) and eigenvalue(s) of an approximate Hamiltonian $\tilde{\mathbf{H}}$
- N must be reduced to less than $\sim 10^9$ by choosing appropriate Slater determinants $\mathbf{D}_i$

CIPSI Algorithm

$$\mathbf{H}\,\boldsymbol{\Psi} = E\,\boldsymbol{\Psi} \quad ; \quad \boldsymbol{\Psi} = \sum_{i=1}^{N_{\text{det}}} \Psi_i \mathbf{D}_i$$

1. Start with $n = 0$, $\boldsymbol{\Psi}^{(n)} = \mathbf{D}_0$
2. Selection : Identify a new set $\mathcal{D}_n = \{\mathbf{D}_i\}$ which are likely to have a large contribution Ψ_i
3. Extend the vector space $(\cup_{m<n}\mathcal{D}_m) \cup \mathcal{D}_n$:
4. Davidson's algorithm : Find the pair $\boldsymbol{\Psi}^{(n+1)}, E^{(n+1)}$, the lowest root of $\mathbf{H}$ in the new vector space
5. Go to 2.

Task-based Parallelism

MPMD : Multiple Program / multiple data

- One executable : the task scheduler
- One executable : the master compute process (OpenMP)
- One/Many executable(s) : slave compute processes (MPI/OpenMP, 1 process/node)
- One process to tunnel data through different networks
- Inter-process communication with ZeroMQ

Task-based Parallelism

Design

Master

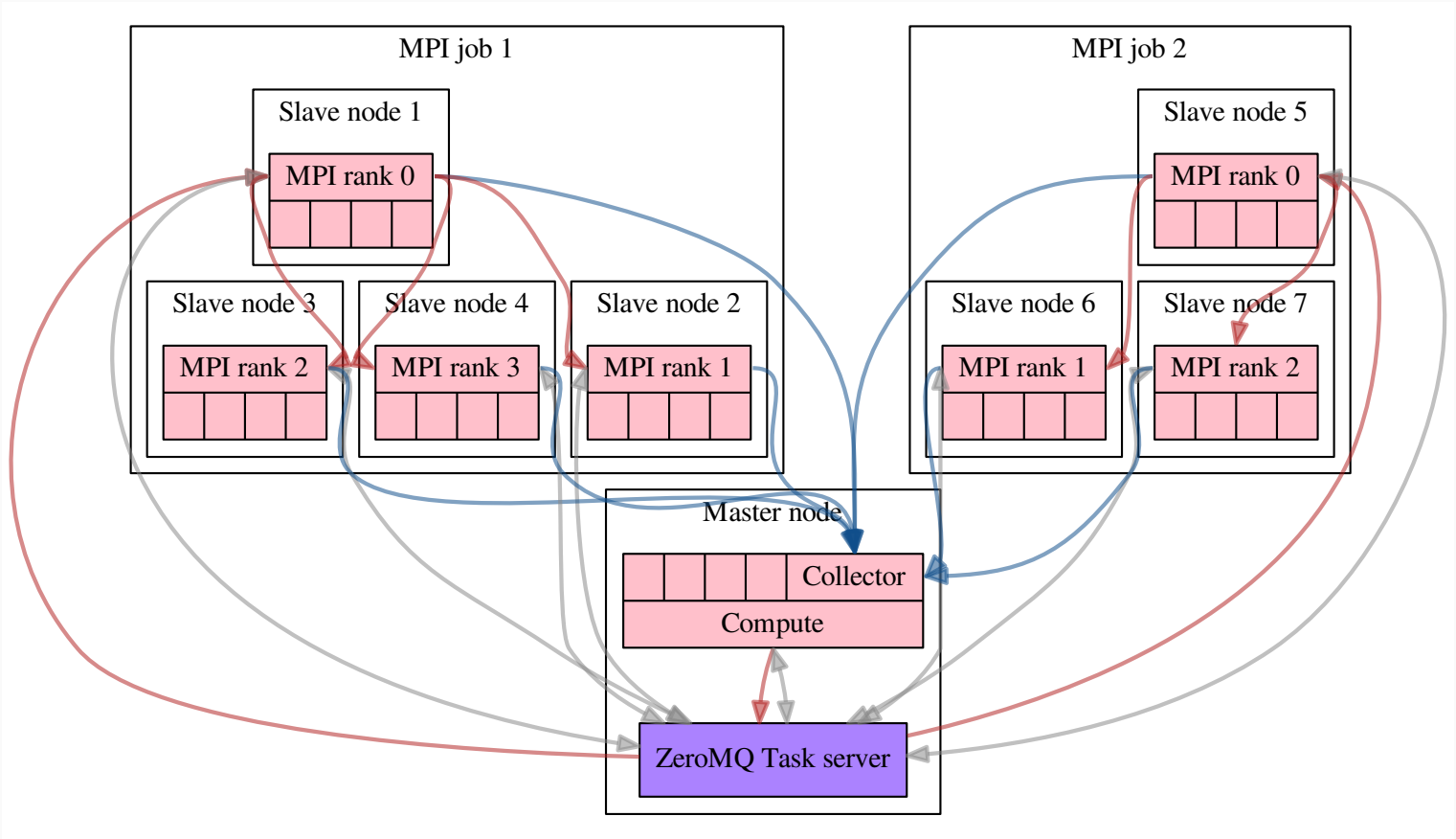
Slave

Tunnel

Motivations

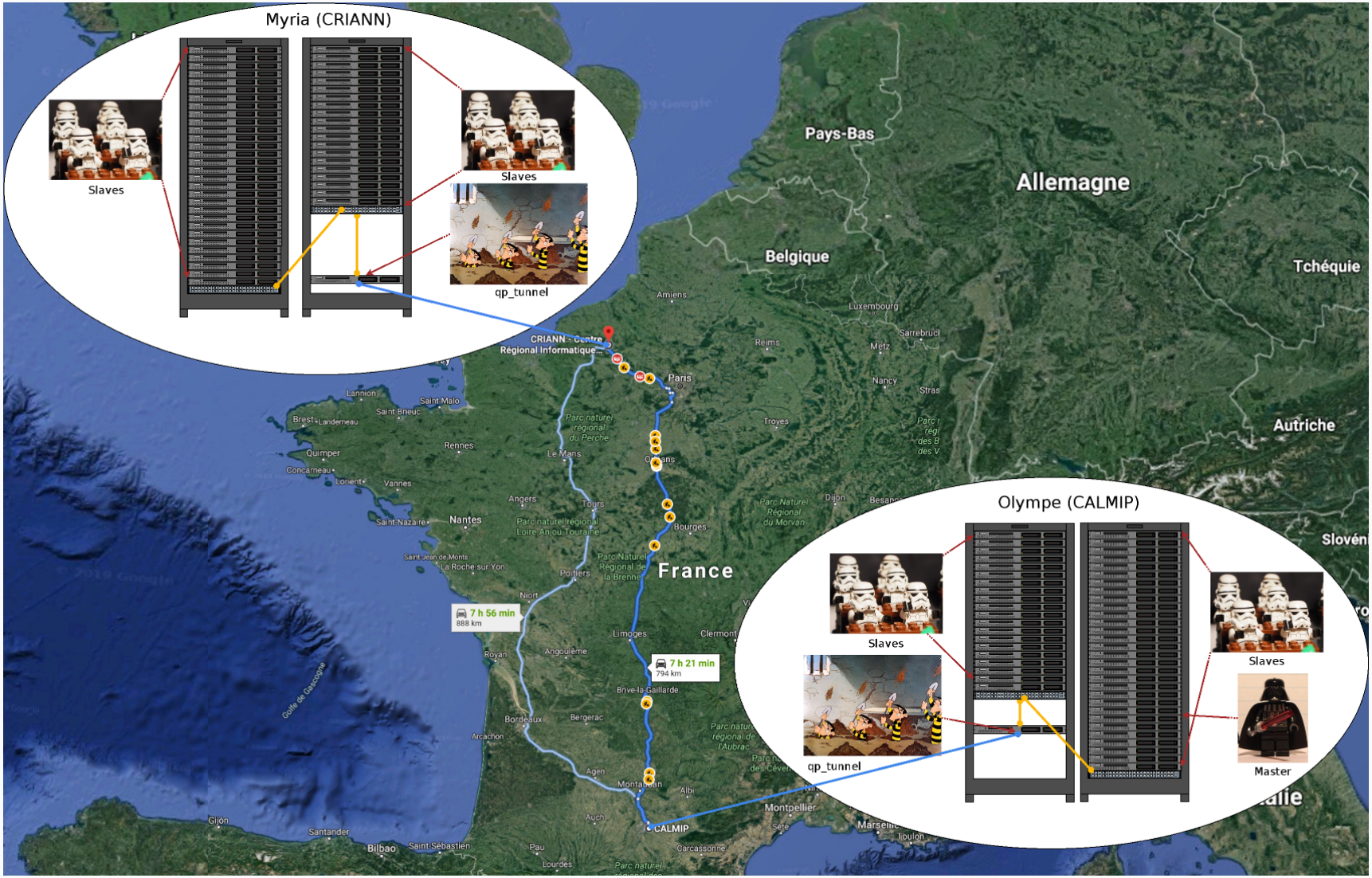
- Possibility to increase dynamically computational resources
- Fault tolerance
- Possibility to use cloud infrastructures

Parallelism



Each task is computed with all possible OpenMP threads.

Experimental setup



Results : Measurements

Bandwidth

CALMIP login $\leftrightarrow$ CALMIP compute	IB EDR 100Gb/s
CRIANN login $\leftrightarrow$ CALMIP login	Renater : 74.1 MB/s
CRIANN login $\leftrightarrow$ CRIANN compute	Omnipath 100GiB/s

Latency (ping)

CALMIP login $\leftrightarrow$ CALMIP compute	0.09 ms
CRIANN login $\leftrightarrow$ CALMIP login	16.72 ms
CRIANN login $\leftrightarrow$ CRIANN compute	0.23 ms

Results : Davidson's diagonalization

- Size of the vectors : $N = 21\,691\,814$, 109 tasks
- 412 MiB sent to each MPI group at the beginning
- 165 MiB sent to each MPI group per Davidson iteration
- 1.5 MiB as a result of a task
- Starting from a bad guess : $[1\ 0\ \ldots\ 0\ 0] \longrightarrow$ 17 iterations

Configuration	N_{CPU}	Wall time
40 nodes Olympe	1440	36:51
40 nodes Myria	1120	44:10
20 nodes Myria, 20 nodes Olympe	1280	43:48

Results : Selection

- Size of the vectors : $N = 21\,691\,814$, $21\,854\,665$ tasks
- Stop when relative error is 0.1% $\longrightarrow \sim 3\%$ of the tasks
- 412 MiB sent to each MPI group at the beginning
- Each task returns 40 bytes
- Each ZeroMQ client fetches m tasks, where m is dynamically adjusted such that the computation of the m tasks takes $\sim N_{\text{CPU}}$ seconds.
- The next m tasks are prefetched during the current computation

Configuration	N_{CPU}	Wall time
50 nodes Olympe	1800	11:58
50 nodes Myria	1400	14:07
25 nodes Myria, 25 nodes Olympe	1600	13:19

Improving Results

Sould we re-try the experiment now?

Quantum Package

https://quantumpackage.github.io/qp2
https://github.com/QuantumPackage/qp2

- Open-source program[1] developed at LCPQ and LCT (Paris)
- Wave function methods for extreme accuracy (benchmarks)
- Computational goal: make wave function methods scalable

[1] Y. Garniron *et al*, *J. Chem. Theory Comput.* **15**, 6, 3591-3609 (2019). https://arxiv.org/abs/1902.08154