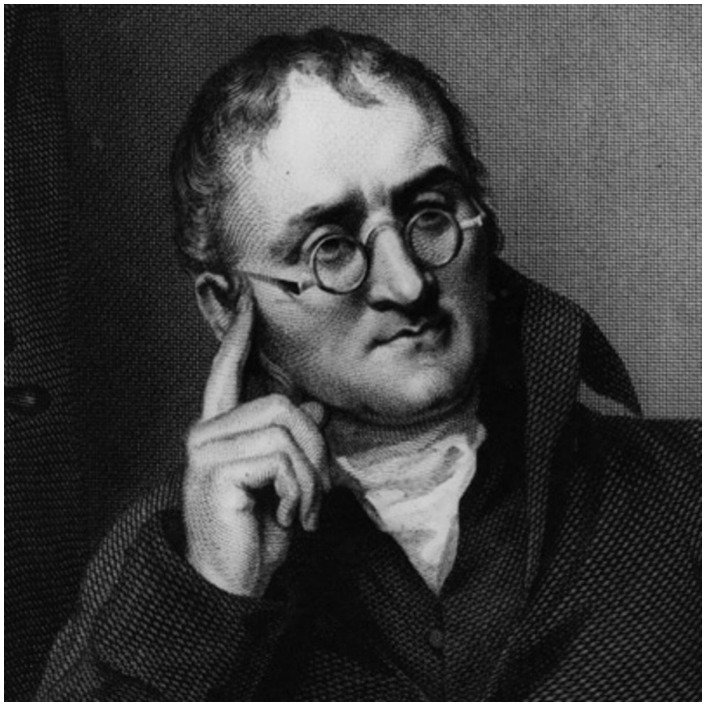
LSDalton on GPUs



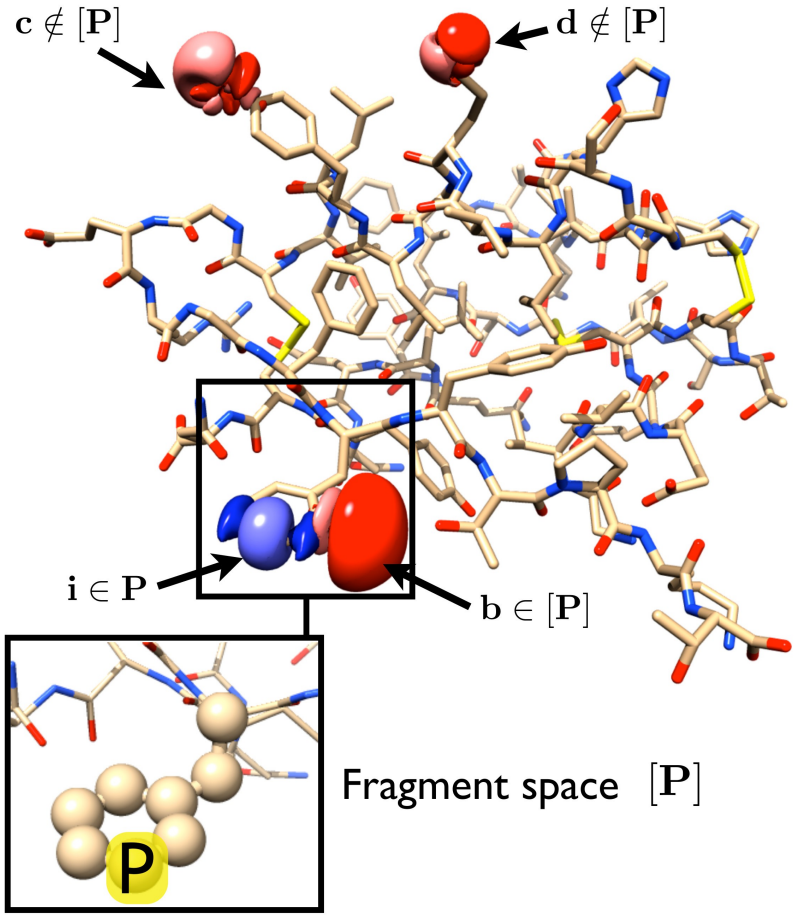
LSDalton, a linear scaling molecular electronic structure program, Release Dalton2015.X (2015), see http://daltonprogram.org.

The DEC-CC module

- Accurate solution of the Schrödinger equation for large molecular systems
- Linear-scaling
- Massively parallel

The CCSD(T) model

- The “golden standard” of quantum chemistry
- Scale as N^7 with system size
- Can be reduced to linear scaling by using locality arguments (DEC-CCSD(T))



GPU-fication

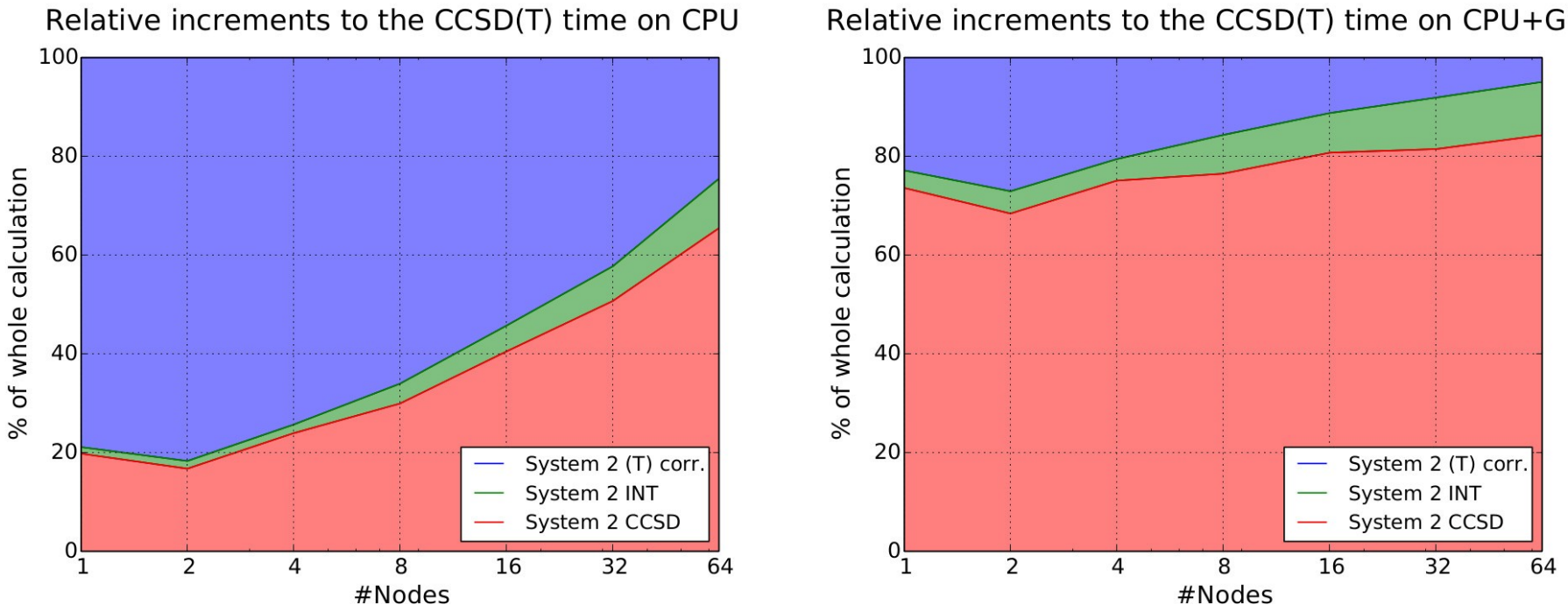


Figure 4. Relative increments to the total time for a CCSD(T) calculation from each of the steps involved; CCSD, integral evaluation (INT) prior to the (T) correction, and the actual (T) correction. The system investigated is system 2.

Profiling

$$\Omega_{ai} = \Omega_{ai}^{A1} + \Omega_{ai}^{B1} + \Omega_{ai}^{C1} + \Omega_{ai}^{D1}$$

$$\begin{aligned}\Omega_{aibj} &= \Omega_{aibj}^{A2} + \Omega_{aibj}^{B2} + P_{ij}^{ab}(\Omega_{aibj}^{C2} + \Omega_{aibj}^{D2} + \Omega_{aibj}^{E2}) \\ &= P_{ij}^{ab}(\tfrac{1}{2}\Omega_{aibj}^{A2} + \tfrac{1}{2}\Omega_{aibj}^{B2} + \Omega_{aibj}^{C2} + \Omega_{aibj}^{D2} + \Omega_{aibj}^{E2})\end{aligned}$$

A2 term:

$$\Omega_{aibj}^{A2} = \tilde{g}_{aibj} + \textstyle\sum_{cd} t_{ij}^{cd}\tilde{g}_{acbd} = \tilde{g}_{aibj} + \tilde{\sigma}_{ij}^{ab}$$

$$\Omega_{aibj}^{A\,2.2} = \sum_{cd} t_{ij}^{cd}\, g_{acbd}$$

B2 term:

$$\begin{aligned}\Omega_{aibj}^{B2} &= \textstyle\sum_{kl} t_{kl}^{ab}\left(\tilde{g}_{kilj} + \sum_{cd} t_{ij}^{cd}\tilde{g}_{kcld}\right) \\ &= \textstyle\sum_{kl} t_{kl}^{ab}\left(\tilde{g}_{kilj} + \sigma_{ij}^{kl}\right) \quad \text{compare to A2 term} \\ &= \textstyle\sum_{kl} t_{kl}^{ab}\zeta_{ij}^{kl}\end{aligned}$$

C2 term:

$$\begin{aligned}\Omega_{aibj}^{C2} &= -\tfrac{1}{2}\textstyle\sum_{ck} t_{jk}^{cb}\left(\tilde{g}_{kiac} - \tfrac{1}{2}\sum_{dl} t_{li}^{ad}\tilde{g}_{kdlc}\right) - \sum_{ck} t_{ik}^{cb}\left(\tilde{g}_{kjac} - \tfrac{1}{2}\sum_{dl} t_{lj}^{ad}\tilde{g}_{kdlc}\right) \\ &= -\tfrac{1}{2}\textstyle\sum_{ck\alpha\gamma} t_{jk}^{cb}\Lambda_{\gamma c}^{h}\Lambda_{\alpha k}^{p}\left(\tilde{g}_{\alpha ia\gamma} - \tfrac{1}{2}\sum_{dl} t_{li}^{ad}\tilde{g}_{\alpha dl\gamma}\right) - \sum_{ck\alpha\gamma} t_{ik}^{cb}\Lambda_{\gamma c}^{h}\Lambda_{\alpha k}^{p}\left(\tilde{g}_{\alpha ja\gamma} - \tfrac{1}{2}\sum_{dl} t_{lj}^{ad}\tilde{g}_{\alpha dl\gamma}\right) \\ &= -\tfrac{1}{2}\textstyle\sum_{\gamma\alpha} t_{j\alpha}^{\gamma b} C'_{\alpha\gamma ia} - \sum_{\alpha\gamma} t_{i\alpha}^{\gamma b} C'_{\alpha\gamma ja} \\ &= \tfrac{1}{2}C_{bjia} + C_{bija}\end{aligned}$$

D2 term:

$$\begin{aligned}\Omega_{aibj}^{D2} &= \tfrac{1}{2}\textstyle\sum_{ck} u_{jk}^{bc}\left[\tilde{L}_{aikc} + \tfrac{1}{2}\sum_{dl} u_{il}^{ad}\tilde{L}_{ldkc}\right] \\ &= \tfrac{1}{2}\left[\textstyle\sum_{ck} u_{jk}^{bc}\tilde{L}_{aikc} + \tfrac{1}{2}\sum_{dl} u_{il}^{ad}\sum_{ck} u_{jk}^{bc}\tilde{L}_{ldkc}\right] \\ &= \textstyle\sum_{\alpha\gamma}\Lambda_{\alpha a}^{p}\Lambda_{\gamma i}^{h}\sum_{ck}\tfrac{1}{2}u_{jk}^{bc}L_{\alpha\gamma kc} + \sum_{dl\alpha\gamma}\tfrac{1}{2}u_{il}^{ad}\Lambda_{\alpha l}^{p}\Lambda_{\gamma d}^{h}\sum_{ck}\tfrac{1}{2}u_{jk}^{bc}L_{\alpha\gamma kc} \\ &= \textstyle\sum_{\alpha\gamma}\Lambda_{\alpha a}^{p}\Lambda_{\gamma i}^{h}D_{\gamma\alpha jb} + \sum_{\alpha\gamma}\tfrac{1}{2}u_{i\alpha}^{a\gamma}D_{\gamma\alpha jb}\end{aligned}$$

E2 term:

$$\begin{aligned}\Omega_{aibj}^{E2} &= \textstyle\sum_{c} t_{ij}^{ac}\left[{}^{I}\tilde{F}_{bc} - \sum_{dkl} u_{kl}^{bd}\tilde{g}_{ldkc}\right] - \sum_{k} t_{ik}^{ab}\left[{}^{I}\tilde{F}_{kj} + \sum_{cdl} u_{lj}^{cd}\tilde{g}_{kdlc}\right] \\ &= \textstyle\sum_{c} t_{ij}^{ac}\left[{}^{I}\tilde{F}_{bc} - H_{bc}\right] - \sum_{k} t_{ik}^{ab}\left[{}^{I}\tilde{F}_{kj} + G_{kj}\right]\end{aligned}$$

A1 and B1:

$$\begin{aligned}\Omega_{ai}^{A1} &= \textstyle\sum_{ckd} u_{ki}^{cd}\tilde{g}_{adkc} = \sum_{p} x_{ap}G_{pi} \\ \Omega_{ai}^{B1} &= -\textstyle\sum_{ckl} u_{kl}^{ac}\tilde{g}_{kilc} = \sum_{q} y_{iq}H_{aq}\end{aligned}$$

C1 and D1:

$$\begin{aligned}\Omega_{ai}^{C1} &= \textstyle\sum_{ck} u_{ik}^{ac}{}^{I}\tilde{F}_{kc} \\ \Omega_{ai}^{D1} &= {}^{I}\tilde{F}_{ai}\end{aligned}$$

- Scale as N^6 with the fragment size
- Can easily take 30% of the time to solution

Compiling

- PGI:
  - Compile with OpenMP, OpenACC, CUBLAS and CUBLAS-XT
  - Cannot link to optimized BLAS libraries
- Cray:
  - CPU version compiles
  - Cannot compile with OpenACC (Internal compiler error, reported)

Porting DGEMMs

- By adapting the code and introduce OpenACC directives
  - Additional tiling to fit tensors on device
  - Hybrid async OpenACC/CUBLAS implementation
  - Compiles (Only PGI)
  - Runs and gives correct results (Thanks Brent!)

- By linking to CUBLAS-XT library
  - Possible to link to any compiler (Thanks Anton!)
  - Compiles (PGI, Cray?)
  - Runs and gives correct results
  - Will be hidden under DGEMM wrapper

```
        !$acc enter data copyin(yv(1:nb*nv),tpl%elm1(1:nor*nvr)) create(w0(1:nb*laleg_req*nv)) async(transp)
        !$acc wait

        !!SYMMETRIC COMBINATION
        ! (w2): I[beta delta alpha gamma] <= (w1): I[alpha beta gamma delta]
        curr_id = 0
        call array_reorder_4d(1.0E0_realk,w1,la,nb,lg,nb,[2,4,1,3],0.0E0_realk,w2)
        do faleg=1,tred,laleg_req

           laleg = laleg_req
           if(tred-faleg+1<laleg_req) laleg = tred-faleg+1

           curr_id = mod(curr_id,nids)+1

           !(w2):I+ [beta delta alpha<=gamma] <= (w2):I [beta delta alpha gamma ] + (w2):I[delta beta alpha gamma]
           call get_I_plusminus_le(w2,'+',fa,fg,la,lg,nb,tlen,tred,goffs,s2,faleg,laleg)
           !(w0):I+ [delta alpha<=gamma c] = (w2):I+ [beta, delta alpha<=gamma] * Lambda^h[beta c]

           !$acc data copyin(w2(1:nb*laleg*nb)) copyout(w3(1+(faleg-1)*nor:nor+(faleg+laleg-2)*nor)) &
           !$acc& async(acc_h(curr_id))

           !call dgemm('t','n',nb*laleg,nv,nb,1.0E0_realk,w2,nb,yv,nb,0.0E0_realk,w0(nb*laleg*nv+1),nb*laleg)
           call ls_dgemm_acc('t','n',nb*laleg,nv,nb,p10,w2,nb,yv,nb,nul,w0,nb*laleg,&
           &i8*nb*laleg*nb,i8*nv*nb,i8*nb*laleg*nv,acc_h(curr_id),cub_h(curr_id))

           !(w2):I+ [alpha<=gamma c d] = (w0):I+ [delta, alpha<=gamma c] ^T * Lambda^h[delta d]
           !call dgemm('t','n',laleg*nv,nv,nb,p10,w0,nb,yv,nb,nul,w2,nv*laleg)
           call ls_dgemm_acc('t','n',laleg*nv,nv,nb,p10,w0,nb,yv,nb,nul,w2,nv*laleg,&
           &i8*laleg*nb*nv,i8*nb*nv,i8*laleg*nv*nv,acc_h(curr_id),cub_h(curr_id))

           !(w0):I+ [alpha<=gamma c>=d] <= (w2):I+ [alpha<=gamma c d]
           call get_I_cged(w0,w2,laleg,nv,acc_h=acc_h(curr_id))

#ifdef VAR_OPENACC
           !(w3.1):sigma+ [i>= j alpha<=gamma] = t+ [c>=d i>=j]^T * (w2):I+ [alpha<=gamma c>=d]^T
           call ls_dgemm_acc('t','t',nor,laleg,nvr,0.5E0_realk,tpl%elm1,nvr,w0,laleg,nul,w3(1+(faleg-1)*nor),nor,&
           &i8*laleg*nvr,i8*nor*nvr,i8*laleg*nor,acc_h(curr_id),cub_h(curr_id))
#else
           !(w3.1):sigma+ [alpha<=gamma i>=j] = (w2):I+ [alpha<=gamma c>=d] * t+ [c>=d i>=j]
           call dgemm('n','n',laleg,nor,nvr,0.5E0_realk,w0,laleg,tpl%elm1,nvr,nul,w3(faleg),tred)
#endif
           !$acc end data
        enddo

        !$acc wait
        !$acc exit data delete(tpl%elm1(1:nor*nvr))
        !$acc enter data copyin(tmi%elm1(1:nor*nvr)) async(transp)
        !$acc wait

#ifdef VAR_OPENACC
        call array_reorder_2d(p10,w3,nor,tred,[2,1],nul,w2)
        call array_reorder_2d(p10,w2,tred,nor,[1,2],nul,w3)
#endif
```

Performances

- Test system:
  - 30s with Cray (CPU with libsci)
  - 140s with PGI (on GPU with CUBLAS and GNU library)
  - 217s with PGI (only CPU with GNU library)

  That's a speed up!!!

  kind of...

- Speed up should increase with the system size

Issues encountered

- Compiling time within supercomputer environments
  - GNU on laptop, no optimization (~4 mins)
  - Cray on Piz daint, no optimization (~30 mins)

- Conflict when linking to libraries