Mise en œuvre de simulations avec le code VASP sur la plateforme Occigen dans une perspective d'utilisation optimale des ressources

Nicole Audiffren,

CINES, Département calcul intensif

Workshop, CINES, 13-14 septembre 2018

Supercalculateur Occigen

- **Partition « HSW » : 2106 nœuds** chacun disposant de 2 processeurs Intel 12-Cores (E5-2690V3@2.6 GHz).
- **Partition « BDW » :1260 noeuds** (chacun disposant de 2 processeurs Intel 14 Cores (E5-2690 V4@2.6GHz)
- **4noeuds de « VISU »** : dotés de processeurs Intel Broadwell (2.6GHz), de 256 Go de RAM, d'une carte Nvidia P100, d'un espace local de stockage temporaire d'environ 700 Go (SSD)
- **1 nœud SKYLAKE**, 2.1 Ghz 224 cores répartis sur 8 sockets, 3To de mémoire DDR4, 2 cartes GPU P100, 12GB, PCIe, espace local temporaire SSD de 3x800 Go

Spécificités de la compilation sur Occigen pour le code VASP

- **Compilateur Intel**
- **Openmpi/intel/2.0.x**
- Clés de précompilation :
  - -Davoidalloc
  - -DMPI_BLOCK=40000 (default=8000)
  - -DscaLAPACK
- Option : **-mkl=sequential**
- FFT : use FFT from MKL library

Spécificités de la compilation sur Occigen pour le code VASP -2

- **MKL_PATH = $(MKLROOT)/lib/intel64**
- **BLAS =**
- **LAPACK =**
- **BLACS = -lmkl_blacs_openmpi_lp64**
- **SCALAPACK = $(MKL_PATH)/libmkl_scalapack_lp64.a $(BLACS)**

- **OBJECTS = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o **
- **$(MKL_PATH)/libfftw3xf_intel.a**
- **INCS =-I$(MKLROOT)/include/fftw**

Spécificités de la compilation sur Occigen pour le code VASP -2

- **MKL_PATH = $(MKLROOT)/lib/intel64**
- **BLAS =**
- **LAPACK =**
- **BLACS = -lmkl_blacs_openmpi_lp64**
- **SCALAPACK = $(MKL_PATH)/libmkl_scalapack_lp64.a $(BLACS)**

- **OBJECTS = fftmpiw.o fftmpi_map.o fftw3d.o fft3dlib.o **
- **$(MKL_PATH)/libfftw3xf_intel.a**
- **INCS =-I$(MKLROOT)/include/fftw**

Libraries dynamiques utilisées

- ***readelf -d vasp_ncl | grep NEEDED***

(NEEDED) Bibliothèque partagée: [libmkl_intel_lp64.so]

(NEEDED) Bibliothèque partagée: [libmkl_sequential.so]

(NEEDED) Bibliothèque partagée: [libmkl_core.so]

(NEEDED) Bibliothèque partagée: [libmkl_blacs_openmpi_lp64.so]

(NEEDED) Bibliothèque partagée: [libstdc++.so.6]

(NEEDED) Bibliothèque partagée: [libmpi_usempif08.so.20]

(NEEDED) Bibliothèque partagée: [libmpi_usempi_ignore_tkr.so.20]

(NEEDED) Bibliothèque partagée: [libmpi_mpifh.so.20]

(NEEDED) Bibliothèque partagée: [libmpi.so.20]

(NEEDED) Bibliothèque partagée: [libm.so.6]

(NEEDED) Bibliothèque partagée: [libpthread.so.0]

(NEEDED) Bibliothèque partagée: [libc.so.6]

(NEEDED) Bibliothèque partagée: [libgcc_s.so.1]

(NEEDED) Bibliothèque partagée: [libdl.so.2]

Méfiez-vous des imitations !

- fichier de configuration pour easybuild

easyconfigs/v/VASP/VASP-5.4.1-intel-2017.4.196-GCC-6.4.0-2.28.eb

remove mkl flag to prevent mixing dynamic libs with the static libs in LIBBLACS/SCALAPACK

prebuildopts += 'sed -i "s|-mkl||" makefile.include && ‘ <<<<<<<<<<<<<<<<<<!!!!!!!!!

- **Vérification**

readelf -d ./build/easybuild/software/VASP.eb_settings/5.4.1-intel-2017b/bin/vasp_std | grep NEEDED

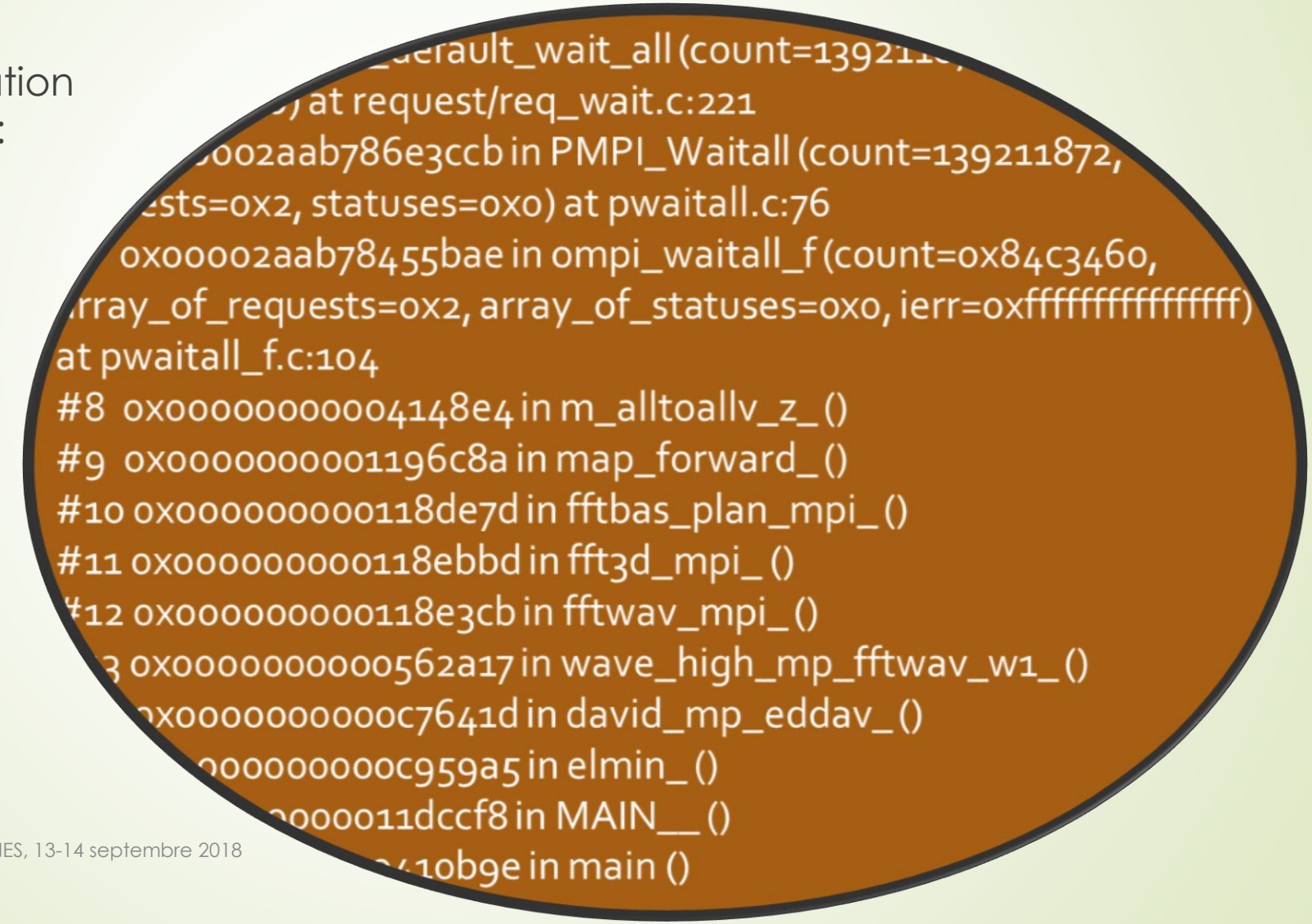
(NEEDED) Shared library: [libmpifort.so.12]
(NEEDED) Shared library: [libmpi.so.12]
(NEEDED) Shared library: [libdl.so.2]
(NEEDED) Shared library: [librt.so.1]
(NEEDED) Shared library: [libpthread.so.0]
(NEEDED) Shared library: [libm.so.6]
(NEEDED) Shared library: [libc.so.6]
(NEEDED) Shared library: [libgcc_s.so.1]

Informations utiles rapportées par la commande *gstack <pid>*

- 0x00002aab786f86b0 in **ompi_coll_base_allreduce_intra_ring** (sbuf=0x84d58c0, rbuf=0x0, count=139287216, dtype=0xffffffffffffffff, op=0xa23be28, comm=0x2aab78fdceb0 <__memcpy_ssse3_back+4576>, module=0x880aba0) at base/coll_base_allreduce.c:462
- #4 0x00002aab786c0fda in PMPI_Allreduce (sendbuf=0x84d58c0, recvbuf=0x0, count=139287216, datatype=0xffffffffffffffff, op=0xa23be28, comm=0x2aab78fdceb0 <__memcpy_ssse3_back+4576>) at pallreduce.c:107
- #5 0x00002aab7844dbf9 **in ompi_allreduce_f** (sendbuf=0x84d58c0 "\001", recvbuf=0x0, count=0x84d5ab0, datatype=0xffffffffffffffff, op=0xa23be28, comm=0x2aab78fdceb0 <__memcpy_ssse3_back+4576>, ierr=0x7ffdea57dbcc) at pallreduce_f.c:87
- #6 0x000000000041732c in m_sum_z_ ()
- #7 0x00000000005ded25 in racc0mu_ ()
- #8 0x00000000005e0cd7 in nonlr_mp_raccmu__ ()
- #9 0x0000000000609cc1 in hamil_mp_hamiltmu_ ()
- #10 0x0000000000c72eec in david_mp_eddav_ ()
- #11 0x0000000000c959a5 in elmin_ ()
- #12 0x00000000011dccf8 in MAIN__ ()

Informations utiles rapportées par la commande *gstack <pid>*

Schéma de communication mis en œuvre par la FFT : all_to_all



OMPI frameworks : Modular Component architecture : MCA

MPI Layer (OMPI) : frameworks destinés à supporter les interfaces MPI avec la couche applicative

Run-time layer (**ORTE**)

- Lauching, monitoring
- Cleaning up tasks in HPC environment

Open Portable Access layer (**OPAL**)

- Portabilité à travers différents operating systems

coll_tuned_use_dynamic_rules=0/1
Switch used to decide if we use static
(compiled/if statements) or dynamic (built at runtime) decision function rules

coll_tuned_alltoall_algorithm
parameter select which alltoall algorithm is used.
It can be locked down to choice of:

0: ignore,
1: basic linear,
2: pairwise,
3: modified bruck,
4: linear with sync,
5: two proc only.

coll_tuned_bcast_algorithm (current value: "ignore »)
Which bcast algorithm is used. Can be locked down to choice of:

0 ignore,
1 basic linear,
2 chain,
3: pipeline,
4: split binary tree,
5: binary tree,
6: binomial tree.

Script batch avec openMPI et tuning des « collectives »

- #!/bin/sh
- #SBATCH --nodes=17
- #SBATCH --constraint=HSW24
- #SBATCH --ntasks-per-node=24
- #SBATCH --mem=118000MB
- #SBATCH --time=24:00:00
- #SBATCH --exclusive
- #SBATCH --output 17nodes.out.**%J**
- module load intel/17.0
- module load openmpi/intel/2.0.1

ulimit -s unlimited

- export **OMPI_MCA_coll_tuned_use_dynamic_rules=1**
- export **OMPI_MCA_coll_tuned_alltoall_algorithm=4**
- export **OMPI_MCA_coll_tuned_bcast_algorithm=6**
- srun --mpi=pmi2 -K1 --resv-ports ./vasp_std

Parallélisation dans VASP

- **NPAR** : parallélisation sur les bandes électroniques. (**Note :** Calculs avec les fonctionnelles hybrides font exception à cette règle car cela exige que NPAR soit alors égal au nombre de coeurs utilisés par k-point - ne pas le specifier alors dans le INCAR).

- **NCORE :** paramètre relié à NPAR via **NCORE=#cores/NPAR**, un seul des deux à préciser.

- **KPAR** : paramètre gouvernant la parallélisation sur les k-points. Il doit être un diviseur de ce nombre de k-points.

- **LPLANE** : parameter booléen qui permet de changer vers une parallélisation sur les PW. Cela réduit la BW pendant la FFT, mais peut créer un déséquilibre entre les taches.

Parallélisation dans VASP

- NPAR groupes de bandes électroniques,
- La parallélisation pour une bande consiste à créer des blocks de PW dans les routines de diagonalization et permet l'étalement de la mémoire
- Quand NPAR augmente, le nombre de PW par coeur augmente ainsi que la mémoire par coeur utilisée.

	Nb de coeurs
FFT	#Nb_coeurs/NPAR
Diagonalisation	NPAR
1 bande	#Nb_coeurs/NPAR

Parallélisation dans VASP, contrôle dans OUTCAR

vasp.5.4.1 24Jun15 (build Jan 31 2017 10:00:26) complex

executed on OCCIGEN date 2018.04.30 14:23:10
running on 408 total cores
distrk: each k-point on 24 cores, **17 groups**

distr: one band on NCORES_PER_BAND= **8 cores**, **3 groups**

KPAR

NPAR

NCORE

NCORE*NPAR=24

Parallélisation dans VASP, contrôle dans OUTCAR

grep NKPTS OUTCAR

k-points **NKPTS = 119** k-points in BZ NKDIM = 119 number of bands **NBANDS= 345**

- Première étape : choisir KPAR comme diviseur du nombre de K-points
- Deuxième étape : fixer NPAR avec une petite analyse de performance pour NPAR de 2 à 4

Cas d'étude : 119 k-points, surface Ti02 (6 couches) catalyseurs Au-Cu

17 nodes et KPAR=17 (119K-points)	HSW	HSW	HSW	HSW
NPAR	8	4	3	2
NCORE	3	6	8	12
memory used on root node (MBytes)	898	888	880	883
max memory (MBytes)	1 829	1 769	1 672	1 606
User time (seconds)	12131	11029	9 621	8815
Elapsed time (seconds)	13305	12302	10 568	9623
Ratio user/elapsed	0,91	0,90	0,91	0,92
node days	2,62	2,42	2,08	1,89
core hours	1507,9	1394,2	1197,7	1090,6
total used cores	408	408	408	408
max nb of PW /node	10153	5 095	3 822	2553
nombre de cores pour la diagonalisation	8	4	3	2
nombre de cores pour la FFT et pour une bande	3	6	8	12

Cas d'étude : 119 k-points , essai sur BDW28

17 nodes et KPAR=17 (119 K-points)	HSW	HSW	BDW	HSW	HSW
NPAR	8	4	4	3	2
NCORE	3	6	7	8	12
memory used on root node (MBytes)	898	888	765	880	883
max memory (MBytes)	1 829	1 769	1 656	1 672	1 606
User time (seconds)	12131	11029	9 785	9 621	8815
Elapsed time (seconds)	13305	12302	10 984	10 568	9623
Ratio user/elapsed	0,91	0,90	0,89	0,91	0,92
node days	2,62	2,42	2,16	2,08	1,89
core hours	1507,9	1394,2	1244,9	1197,7	1090,6
total used cores	408	408	476	408	408
max nb of PW /node	10153	5 095	4 373	3 822	2553
nombre de cores pour la diagonalisation	8	4	4	3	2
nombre de cores pour la FFT et pour une bande	3	6	7	8	12

Cas d'étude avec HSE06

HSE06	56 K-points	
	HSW	HSW
#nodes	**27**	56
KPAR	**27**	56
NPAR	24	24
NCORE	1	1
memory used on root node (MBytes)	94	92
max memory (MBytes)	525	533
User time (seconds)	7018	3149
Elapsed time (seconds)	7116	3352
Ratio user/elapsed	0,99	0,94
node days	**2,22**	**2,17**
core hours	1280,9	2656,6

Cas d'étude comportant un grand nombre de bandes electroniques ~1458 , 7 k-points, 300 atomes

	Max memory per task (MB)	core-hours		KPAR	NPAR	efficacité
20 nodes	1618	559		5	6	1
25 nodes	1287	518		5	6	0,84
35 nodes	985	232		7	6	0,84

« D'autres situations demandant plus de mémoire par tâche »

Exceeded job memory limit
slurmstepd: Step 5035165.0 exceeded memory limit (61605648 > 60416000), being killed
slurmstepd: Exceeded job memory limit at some point.

			metaGGA		
417 kpoints	304 bandes électroniques				
1,98 GB root node			KPAR=7		
			NPAR=4		
			28 nodes	mem=118000MB	

- #SBATCH –mem=118000MB

nœuds Haswell seulement

« Best Practices »

Calcul pure DFT

Le meilleur : KPAR = maximal, et NPAR minimal,

Le plus mauvais : KPAR = 1 et NPAR = maximal

Calcul Hybride HF -DFT

Préciser seulement KPAR,

Ne pas specifier NPAR, (automatiquement NCORE=1)

« Best Practices »

Demande DARI

Travail préparatoire nécessaire :

Faire une étude préalable sur la scalabilité et la mémoire requise pour chaque type de calcul

Bénéfice : Demande mieux étayée

Ajouter un titre de diapositive - 1