Computational Platforms for VASP

Robert LORENZ

Institut für Materialphysik and Center for Computational Material Science

Universität Wien, Strudlhofgasse 4, A-1090 Wien, Austria

Outline

- Getting Started
- installation of VASP
- performance comparisons
- parallel LINUX clusters
- running VASP efficiently on a parallel machine

Getting Started - checklist

1. download VASP and unpack VASP
   ftp://cms.mpi.univie.ac.at
2. check your Hardware
   - a. a minimum of 256 MB
   - b. Athlon XP, P4 or better
   - c. HD 250 MB free
3. check your Compiler and Libraries
   - a. FORTRAN 90 compiler
   - b. BLAS, LAPACK
   - c. parallel: MPI software
4. compile and install VASP
5. test vasp

download VASP

a really easy task:

1. get your license `username` and `password`
   (a) one account for vasp and standard potentials
   (b) another account for paw potentials (GGA and LDA)
2. use the vasp account to connect to ftp://cms.mpi.univie.ac.at
   (a) download latest vasp, vasp.lib and vasp.util
   (b) download potentials pot and pot_GGA
3. use the paw acount to connect to ftp://cms.mpi.univie.ac.at
   (a) download GGA and LDA potential files

content of ˜`vasp`

```
  bin   executable  incoming  log  pot_GGA  src
 beta   doc  html       lib       pot  script   tmp
```

download VASP (cont.)

we concentrate on the following directories:

`~vasp/src`	containes the source in `.tar.gz` and `.Z` format,
	download the latest versions of vasp.X.Y and vasp.X.lib
	X *ldots* major version, Y *ldots* minor version
`~vasp/doc`	some useful documentation, check also
	`http://cms.mpi.univie.ac.at`

installation of VASP

minimum requirements

1. FORTRAN90 Compiler
2. BLAS (recommended machine optimized - atlas, cxml, mkl, ...)
3. MPI library for parallel execution (lam, mpich, ...)

```
mkdir vasp; cd vasp
tar xvzf vasp.4.lib.tar.gz
tar xvzf vasp.4.5.tar.gz
```

- create a directory
- untar the lib and vasp source

installation of VASP (cont.)

```
cd vasp.4.lib

cp makefile.linux_ifc_P4 makefile

make

cd ../vasp.4.5

cp makefile.linux_ifc_P4 makefile

make
ls -l vasp
```

- start making the vasp library
- choose a predefined makefile
- make the vasp library
- let’s make vasp
- choose a predefined makefile
- make vasp (long task)
- check if vasp really exists … ;-)

installation of VASP (cont.)

check your vasp installation

```
mkdir bench; cd bench
ftp://vasp@cms.mpi.univie.ac.at
~/src/benchmark.tar.gz


tar xvzf benchmark.tar.gz
vi INCAR
IALGO = 48

~/vasp/vasp.4.5/vasp
```

- make the directory bench
- get the file `benchmark.tar.gz`
- untar `benchmark.tar.gz`
- change the `INCAR` file
  line with `IALGO = 8` $\rightarrow$ `IALGO = 48`
  (recent software patent issue)
- start vasp

installation of VASP (cont.)

result of benchmark

```
...
RMM:   4    -0.902263862459E+04   -0.13123E-04   -0.38949E-05    49   0.351E-02
5 T=  1918. E= -.90209004E+04 F= -.90226386E+04 E0= -.90218782E+04  EK= 0.17355E+01
                                                                      SP= 0.26E-02 SK= 0.94E-04

diff OSZICAR OSZICAR.ref_4.4.3
< DAV:   3    -0.902263881286E+04   -0.17813E-05   -0.17808E-05    72   0.232E-02    0.284E-02
< DAV:   4    -0.902263882471E+04   -0.11847E-04   -0.88141E-06    56   0.459E-02
< 5 T=  1918. E= -.90209007E+04 F= -.90226388E+04 E0= -.90218783E+04  EK= 0.17354E+01
                                                                      SP= 0.26E-02 SK= 0.94E-04
---
> CG :   1    -0.902264199383E+04   -0.90226E+04   -0.25955E-03    99   0.166E-01    0.241E-02
> CG :   2    -0.902264200818E+04   -0.14350E-04   -0.45941E-05    49   0.354E-02
> 5 T=  1918. E= -.90209037E+04 F= -.90226420E+04 E0= -.90218816E+04  EK= 0.17356E+01
                                                                      SP= 0.26E-02 SK= 0.94E-04
```

- test runs 20s on a fast Pc
- small differences to the reference file `OSZICAR.ref_4.4.3` are allowed

Platforms for VASP

makefiles in vasp.4.5

```
makefile.cray      makefile.hp            makefile.linux_ifc_P4
makefile.rs6000    makefile.sun           makefile.dec
makefile.linux_abs makefile.linux_ifc_ath makefile.rs6000_p1
makefile.t3d       makefile.fujitsu       makefile.linux_alpha
makefile.linux_pg  makefile.sgi           makefile.t3e
makefile.gen       makefile.linux_ifc     makefile.nec
makefile.sp2       makefile.vpp
```

cray	CRAY C90, J90, T90
dec	DEC ALPHA, True 64 Unix
hp	HP PA
linux_abs	Linux, Absoft compiler
linux_alpha	Linux, Alpha processors fort compiler
linux_ifc_P4	Linux, Intel fortran compiler (ifc), P4 optimisation
linux_ifc_ath	Linux, Intel fortran compiler (ifc), Athlon optimisation
linux_pg	Linux, Portland group compiler
nec	NEC vector computer
rs6000	IBM AIX, xlf90 compiler
sgi	SGI, Origin 200/ 2000/ 3000, Power Challenge, O2 etc.
sp2	IBM SP2, possibly also usefull for RS6000
sun	SUN, Ultrasparc
t3d	Cray/SGI T3D
t3e	Cray/SGI T3E
vpp	fujitsu VPP, VPX

Platforms for VASP (cont.)

benchmark settings used

1. *bench_Hg* 50 Hg atoms, empty core PP, 1 kpoint
   `NBANDS=316; ENMAX = 140 eV;ISYM = 0`
   around 5 minutes on a fast Pc
2. *bench_PdO* 75 Pd and 12 O atoms = 87 atoms, 5x4 supercell, 1 kpoint
   `NBANDS=496; ENMAX = 250 eV`
   1.5 hours on a fast Pc

Platforms for VASP (cont.)

single cpu systems

system	bench Hg	bench PdO
IBM SP3 HN	356	
IBM SP4	181	4000
HP ES45 1GHz	256.23	5326.31
P4 2.53GHz P4T533	271.04	5676.65
P4 2.53GHz P4G8X	293.39	5823.43
Xeon 2.4GHz i7505	294.03	6160.10
AMD XP 1700+ a7m266	504.50	
CRAY T3E 1200	420.00	
P4 2.8 GHz P4SAA exp	265.03	5481.72

Platforms for VASP (cont.)

important hardware parameter

cpu	the cpu throughput is very important for the vasp performance, many time consuming routines in vasp use BLAS/LAPACK, streaming methods give an enormous performance boost (Intel's SSE2)
memory	vasp requires 512 - 1024 MByte / cpu; the computational speed of vasp depends on the sustained memory bandwidth
HD	non critical, large enough to hold `WAVECAR` and `CHG*`

Performance

some remarks

- Clusters of PCs are an attractive platform for parallel applications because of their cost effectiveness
- VASP supports parallel execution on commodity components using MPI for parallel communication

performance benchmarks

using vasp as a benchmark

- always use the same vasp version, otherwise the timings are not comparable
- use the best compiler options and the fastest available blas for the test platform

Performance (cont.)

some remarks

- How to improve performance?
  - on the supported platforms we have included high performing settings (makefile, algorithm)
  - new compiler & new hardware:
    - ∗ start from a makefile for a similar system
    - ∗ check compiler flags
    - ∗ use vendor BLAS/LAPACK or generate a highly optimized atlas
    - ∗ try different `CACHE_SIZE` settings
- different results on different hardware/software
  - small differences caused by:
    - ∗ different implementations of BLAS/LAPACK
    - ∗ different optimizations of the compiler
    - ∗ different behavior of the FPU (pc80, rounding)

  - large errors and/or crashes
    - ∗ hardware failure (check your memory)
    - ∗ too high optimization

- vasp running on system XXX ?
  - FORTRAN 90 compiler supporting the standard F90 language
  - BLAS/LAPACK `http://www.netlib.org` it's free, but usually slower than vendor supplied
  - some scripting language
  - parallel execution: standard MPI V1 implementation

parallel LINUX clusters

Hardware

- Beowulf: PC cluster with commodity components
- fast but cheap network

Software

- Message Passing Interface (MPI), Open-Source (lam, mpich)
- scheduler with parallel support
- cluster administration

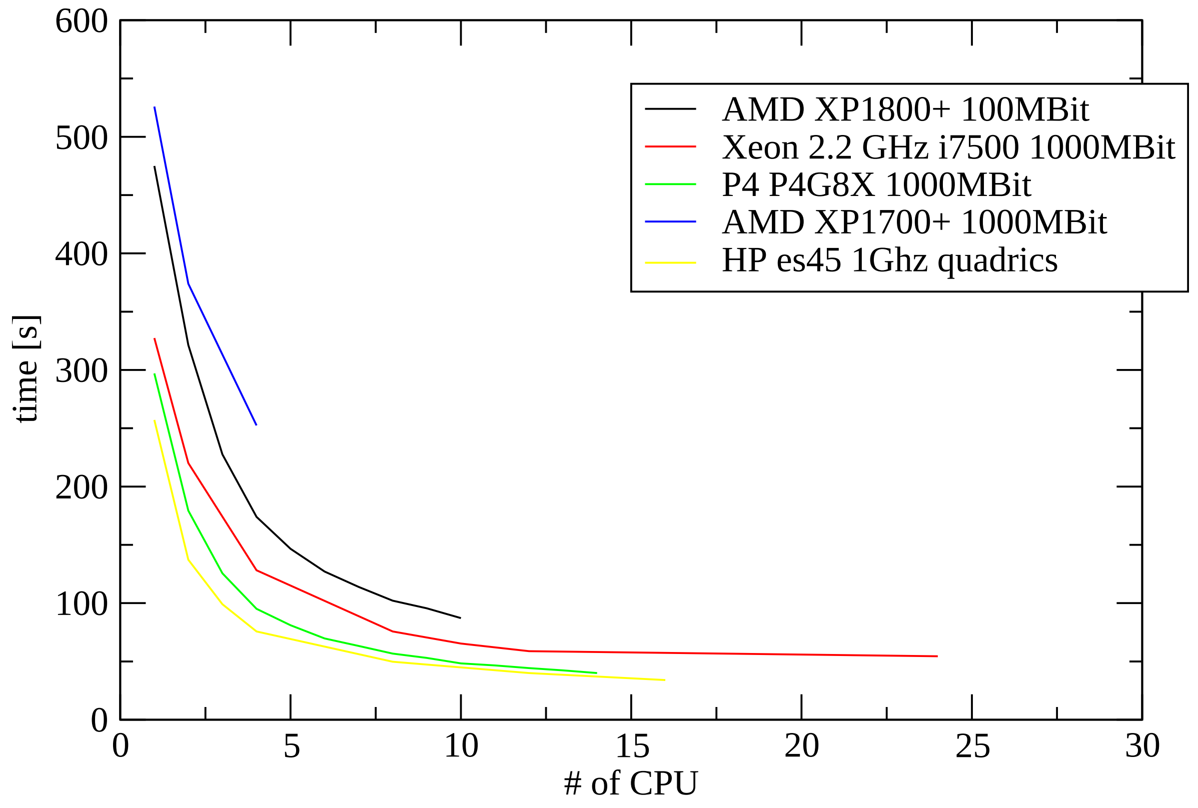
parallel LINUX clusters

some important notes

```
FC=mpif77
CPP    = $(CPP_) -DMPI \
         ....
        -DNGZhalf -DMPI_BLOCK=2000
         ....
FFT3D   = fftmpi.o fftmpi_map.o \
          fft3dlib.o
```

- use the wrapper from your MPI installation as compiler
- `-DMPI` uses the vasp mpi support
- use `-DNGZhalf` instead of `-DNGXhalf`
  the MPI blocksize influences the communication time on the MPI net
- vasp supports lam and mpich MPI implementations, but lam is preferred

parallel LINUX clusters (cont.)



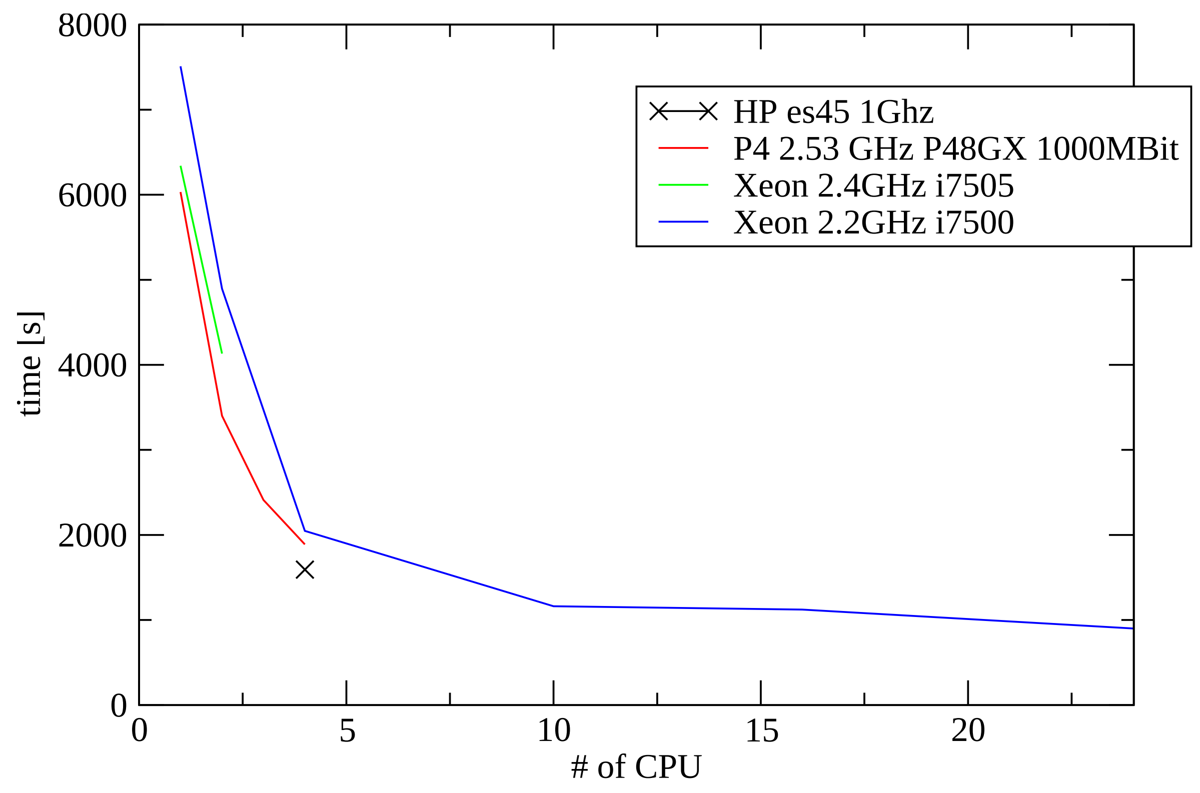
benchmark: bench_Hg

small system

# of CPU's > 8

Latency bound

parallel LINUX clusters (cont.)



benchmark: bench_PdO

large system

# of CPU's > 16

Latency bound

parallel LINUX clusters (cont.)

important parameters for cluster running vasp

- **network**
  - **latency bound** gives a lower bound for the MPI communication blocksize. limits the maximum number of nodes. a typical Ethernet based network (100MBit, 1000MBit) shows around 90μs latency. optimizations on modern Gigabit cards reduce the latency to 30μs
  - **bandwidth bound** the maximum transfer rate limits the communication speed. vasp uses all–to–all communication $\rightarrow$ collision bound
- **cpu**
  - overall performance is limited by the network efficiency $\rightarrow$ increasing the cpu speed alone in not enough
  - fast data-path from network device to cpu (memory)

Amdahl's Law 1967

speedup:

$s = \frac{t(1)}{t(N)}$ $t(1)$ $ldots$ serial time $t(N)$ $ldots$ time for N CPU's

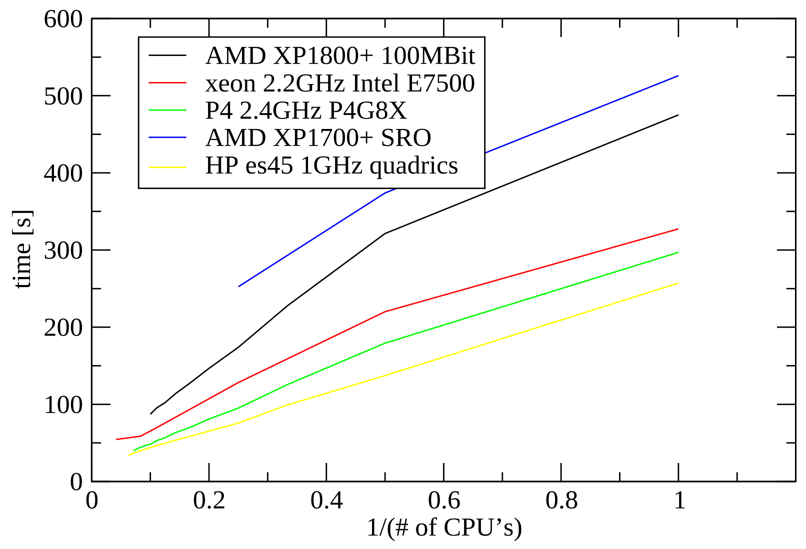
Amdahl's Law

$s = \frac{N}{(B \times N) + (1 - B)}$ $B \ldots$ % of algorithm that is serial

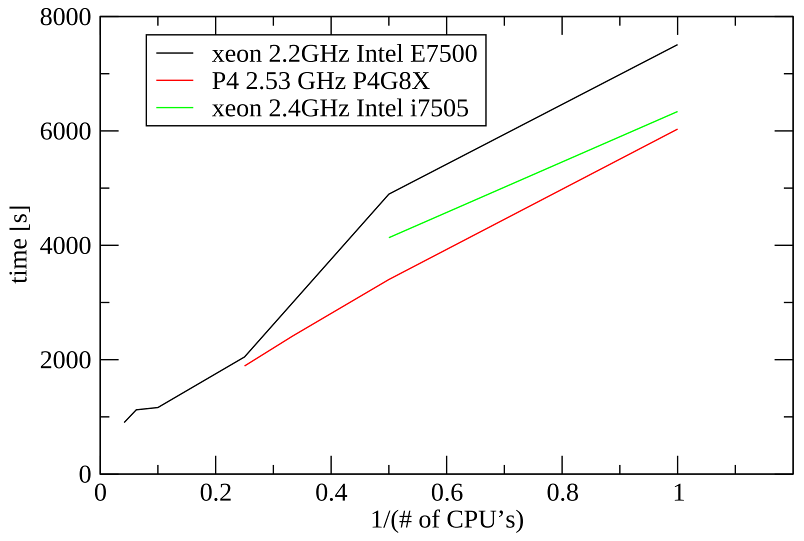
$\rightarrow$ it exists an upper limit on the number of CPU's for a given problemsize

$\rightarrow$ scale the problem with the numer of CPU's

$\rightarrow$ use fast Computers with a good price performance ratio

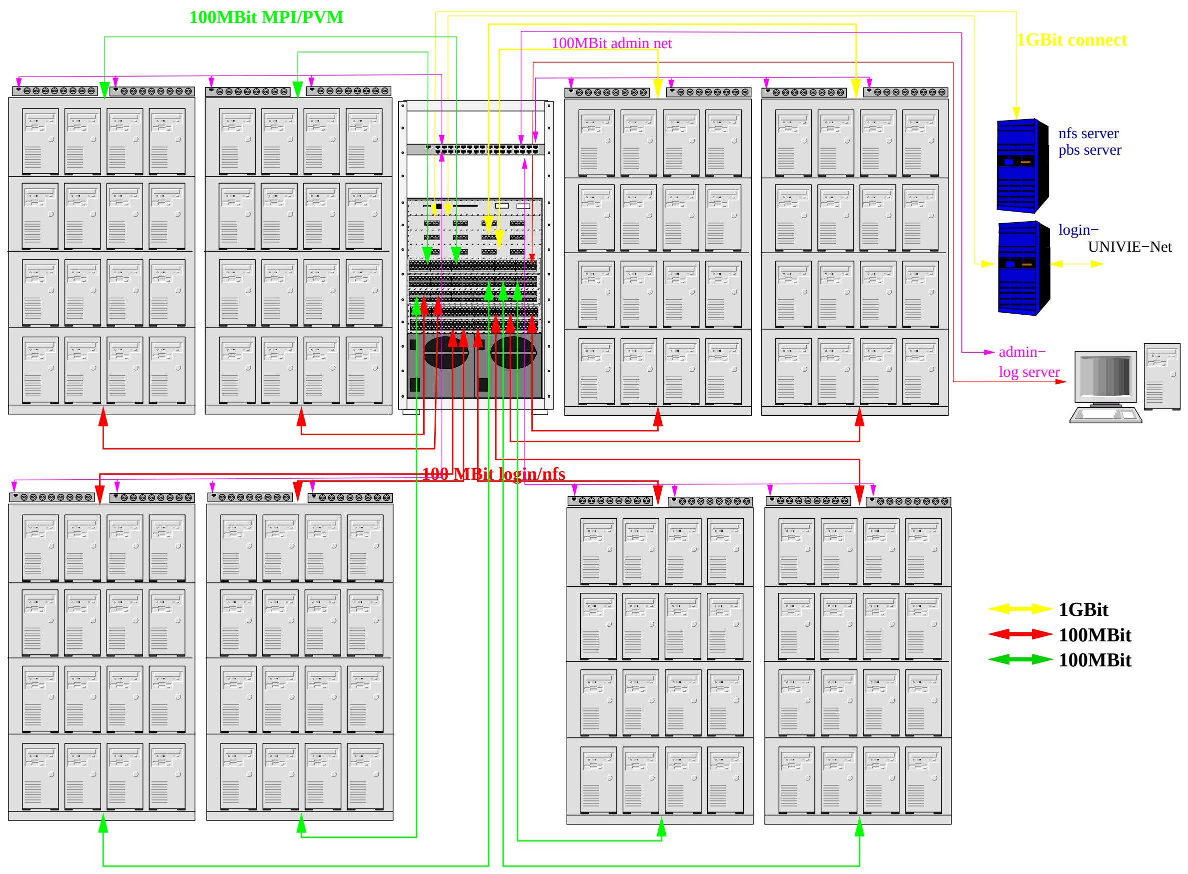


small bench_Hg



big bench_PdO

a Linux Cluster for Vasp



Schrödinger I