HPC SAV Software Stack and Efficacy

Efficacy as the centre point

Ot(t)o Kohulák

SAV BA

June 10, 2026

Outline

Devana

- Login Nodes
- 2x Intel 32 cores
- 2x 1x A100 40GB
- x86_64
- Login[1-2]

- CPU Compute Nodes
- 2x Intel 32 cores
- 0x A100 40GB
- x86_64
- n[001-140]

- GPU Compute Nodes
- 2x Intel 32 cores
- 8x 4x A100 40GB
- x86_64
- n[141-148]

Perun

- Login Nodes
- 2x AMD Epyc 160 cores
- No GPU
- x86_64
- Login[1-4]

- CPU Compute Nodes
- 2x AMD Epyc 160 cores
- No GPU
- x86_64
- cn[001-060]
- cn[045-060] have twice of RAM

- 4x GH200 - each with 72 Neoverse ARM cores and 210GB RAM and 120 GB GPU RAM (96GB usable)
- ARM aarch64
- gn[001-078]

Compiling - Perun

```
smc@login03:~$ gcc main.c -o exe_cpu
smc@login03:~$ sudo su - okohulak
Last login: Tue Jun  9 13:12:01 CEST 2026 on pts/3
[okohulak@login03 ~]$ mkdir example
[okohulak@login03 ~]$ vim main.c
[okohulak@login03 ~]$ gcc main.c -o exe_cpu
[okohulak@login03 ~]$ cat main.c
#include <stdio.h>

int main(void) {
    int a = 5;
    int b = 7;
    int sum = a + b;

    printf("Hello from C!\n");
    printf("%d + %d = %d\n", a, b, sum);

    return 0;
}
[okohulak@login03 ~]$
```

```
[okohulak@gn001 ~]$ gcc main.c -o exe_gpu
[okohulak@gn001 ~]$ cat main.c
#include <stdio.h>

int main(void) {
    int a = 5;
    int b = 7;
    int sum = a + b;

    printf("Hello from C!\n");
    printf("%d + %d = %d\n", a, b, sum);

    return 0;
}
[okohulak@gn001 ~]$
[okohulak@gn001 ~]$
[okohulak@gn001 ~]$
[okohulak@gn001 ~]$
[okohulak@gn001 ~]$
[okohulak@gn001 ~]$
[okohulak@gn001 ~]$
[okohulak@gn001 ~]$
```

Executing - Perun

```
[okohulak@gn001 ~]$ file exe_gpu
exe_gpu: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=330fe7ca2d5856754577b8c7130c28e0d394e208, for GNU/Linux 3.7.0, not stripped
[okohulak@gn001 ~]$ file exe_cpu
exe_cpu: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=66f68fb3fa355c10e332bbe2b390a89d73b45c83, for GNU/Linux 3.2.0, not stripped
```

Executing - Perun

```
[okohulak@gn001 ~]$ file exe_gpu
exe_gpu: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=330fe7ca2d5856754577b8c7130c28e0d394e208, for GNU/Linux 3.7.0, not stripped
[okohulak@gn001 ~]$ file exe_cpu
exe_cpu: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=66f68fb3fa355c10e332bbe2b390a89d73b45c83, for GNU/Linux 3.2.0, not stripped
```

```
[okohulak@gn001 ~]$ ./exe_cpu
```

Executing - Perun

```
[okohulak@gn001 ~]$ file exe_gpu
exe_gpu: ELF 64-bit LSB executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-aarch64.so.1, BuildID[sha1]=330fe7ca2d5856754577b8c7130c28e0d394e208, for GNU/Linux 3.7.0, not stripped
[okohulak@gn001 ~]$ file exe_cpu
exe_cpu: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=66f68fb3fa355c10e332bbe2b390a89d73b45c83, for GNU/Linux 3.2.0, not stripped
```

```
[okohulak@gn001 ~]$ ./exe_cpu
```

```
bash: ./exe_cpu: cannot execute binary file: Exec format error
```

Python

- I discourage using Conda - it contaminates your home directory and can cause issues with the system Python
- Use virtualenv or venv instead
- Always have environemnt for CPU partition and one for GPU partition
- When you load Python on CPU partition you gen a different Python than on GPU partition - they are not the same and have different packages installed

Software Stack eXchange

- Login nodes and compute nodes are the same architecture and they see the same software stack
- GPU nodes are different, they have different softeware stack
- One can switch between GPU and CPU software stack using utility `ssx`: `ssx set_gpu`
- You can load modules from GPU software stack but you cannot execute them on CPU nodes, Login nodes and vice versa
- Unfortunatelly you cannot interactively work on ARM architecture as you would do on x86_64 with Login nodes. You have to submit a job to GPU partition and then you can work interactively on ARM architecture.

```
srun -G1 -N1 -A <account> -p gpu_short -pty /bin/bash
```

Amdahl's Law

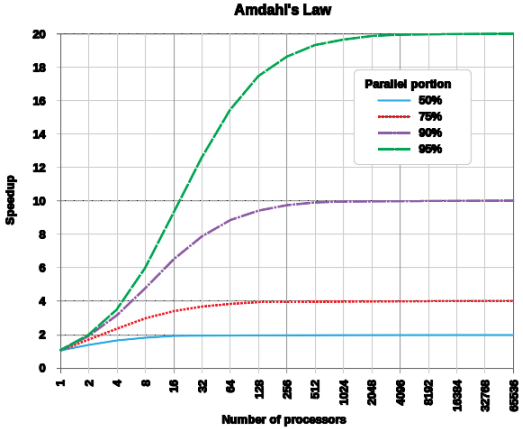
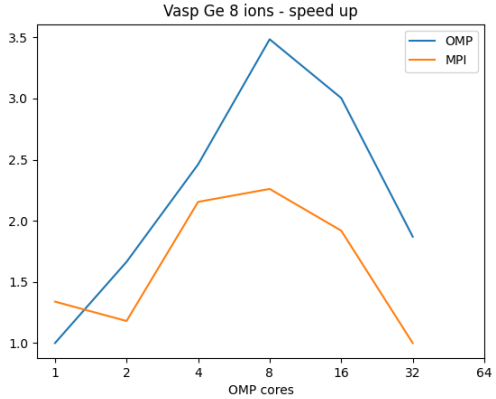
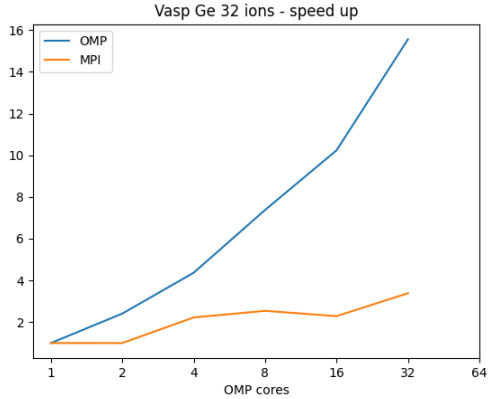


Figure: Amdahl's Law - Wikipedia

Efficacy

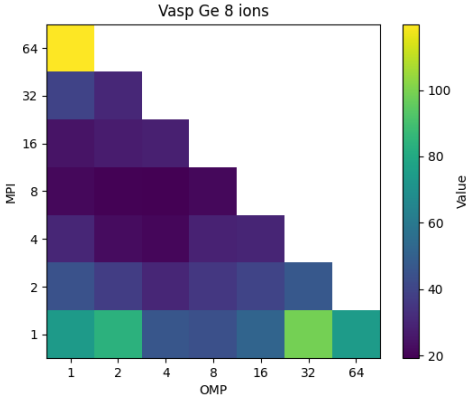

Vasp Ge 8 ions - speed up
OMP
MPI
3.5
3.0
2.5
2.0
1.5
1.0
1
2
4
8
16
32
64
OMP cores

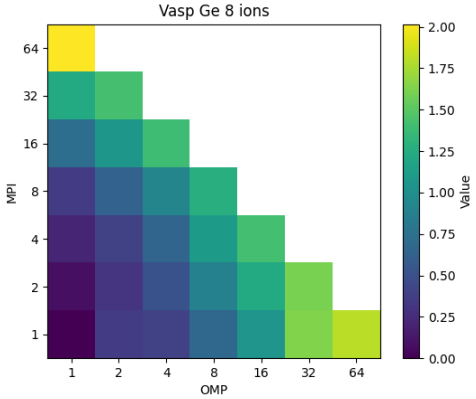

Vasp Ge 32 ions - speed up
OMP
MPI
16
14
12
10
8
6
4
2
1
2
4
8
16
32
64
OMP cores

Efficacy

Vasp Devana

$$\text{Efficacy} = \log_{10}\left(\frac{\text{Time}}{\text{Time fastest}} * \text{processes} * \text{cpus}\right)$$

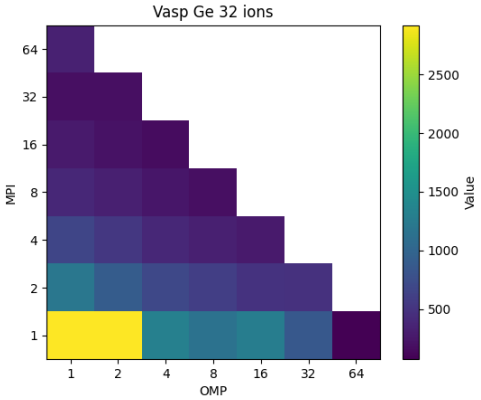


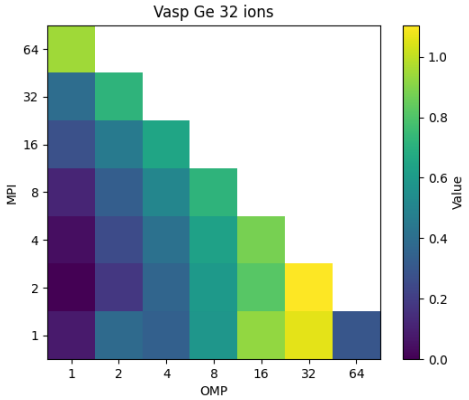


Efficacy

Vasp Devana

$$\text{Efficacy} = \log_{10}\left(\frac{\text{Time}}{\text{Time fastest}} * \text{processes} * \text{cpus}\right)$$





Perun -Efficacy

CPU partition

- If you know how to work on Devana, you will be able to work on Perun CPU partition as well
- You have much more cores, use them wisely
- You dont have to use as many nodes as you would on Devana, you can use more cores per node and less nodes
- You dont have to use the whole node, if it is not necessary, use your - ours resources wisely

GPU partition

- Different software stack, different architecture, different way of working
- Less software available
- State-of-the-art hardware, today is hard to find much better nodes that these ones
- Containerization is your friend not foe
- Not best for running inferences, but best for training

Takeaways

- Think before compute
- HW topology matters
- Software stack matters
- Do some tests before you run your production job
- It is new system, new architecture, new challanges, be patient and open minded
- Encountering problems? - ask us, we are here to help you

Any problems? Reach us on our ticketing system.

Thank you!

Questions?