Using Supercomputers for PIC

Sigvald Marholm

University of Oslo
Department of Physics

01.02.22

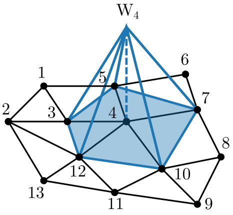
Outline

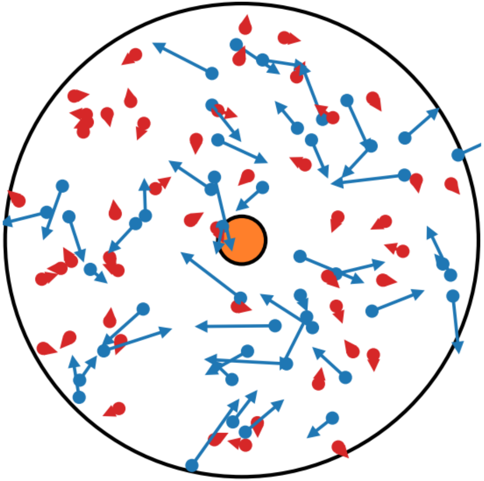
Outline

Recap: the Particle-In-Cell Method

PIC cycle:

1. Weight charge from particles to mesh
2. Solve Poisson equation ($\rho \rightarrow \mathbf{E}$)
3. Weigth field from mesh to particle
4. Move particles ($m\ddot{\mathbf{x}} = q\mathbf{E}$)

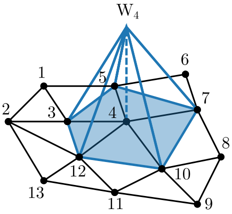


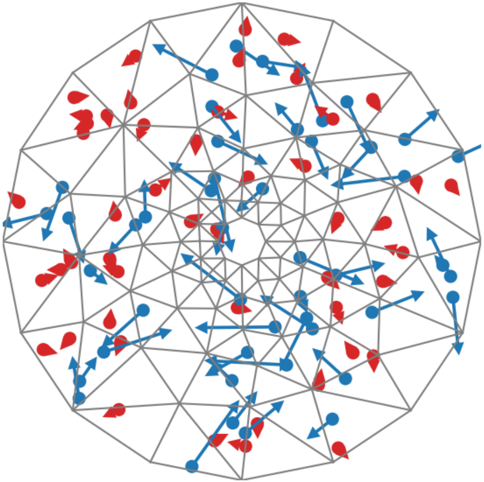


Recap: the Particle-In-Cell Method

PIC cycle:

1. Weight charge from particles to mesh
2. Solve Poisson equation ($\rho \rightarrow \mathbf{E}$)
3. Weigth field from mesh to particle
4. Move particles ($m\ddot{\mathbf{x}} = q\mathbf{E}$)

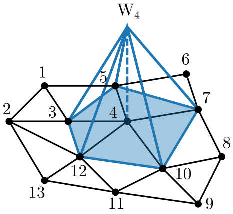


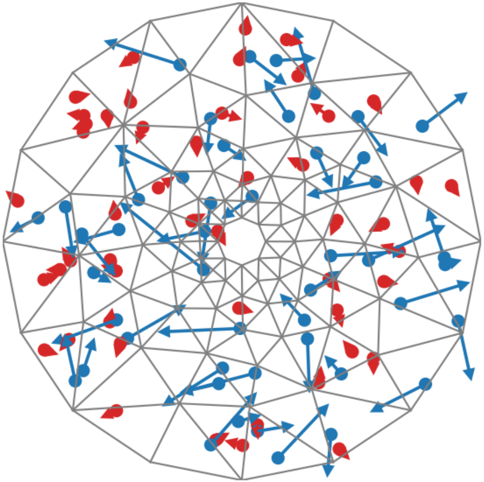


Recap: the Particle-In-Cell Method

PIC cycle:

1. Weight charge from particles to mesh
2. Solve Poisson equation ($\rho \rightarrow \mathbf{E}$)
3. Weigth field from mesh to particle
4. Move particles ($m\ddot{\mathbf{x}} = q\mathbf{E}$)

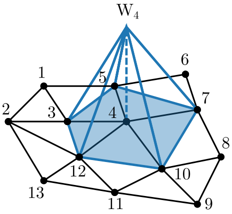


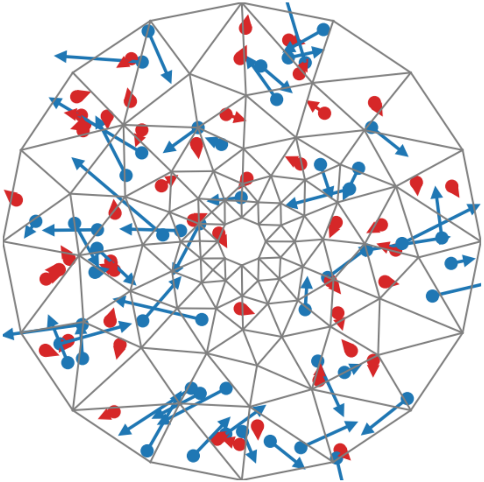


Recap: the Particle-In-Cell Method

PIC cycle:

1. Weight charge from particles to mesh
2. Solve Poisson equation ($\rho \to \mathbf{E}$)
3. Weigth field from mesh to particle
4. Move particles ($m\ddot{\mathbf{x}} = q\mathbf{E}$)

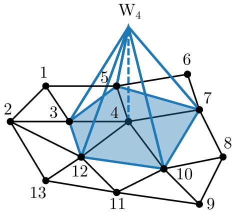




Recap: the Particle-In-Cell Method

PIC cycle:

1. Weight charge from particles to mesh
2. Solve Poisson equation ($\rho \rightarrow \mathbf{E}$)
3. Weigth field from mesh to particle
4. Move particles ($m\ddot{\mathbf{x}} = q\mathbf{E}$)



Recap: important criteria

Spatial (in Gmsh):

$$h \lesssim 3\lambda_{De}$$

$h \sim r/5$ (where r is radius of curvature)

Temporal (automatic in PTetra):

$$\Delta t \ll \omega_{pe}^{-1} \quad (\Delta t < 1.62\omega_{pe}^{-1})$$

$$\Delta t \ll \omega_{ge}^{-1}$$

$\Delta t < v_p^{-1} h$ for "most" particles p

$(\Delta t < kc^{-1} h$ for EM codes, for some $k)$

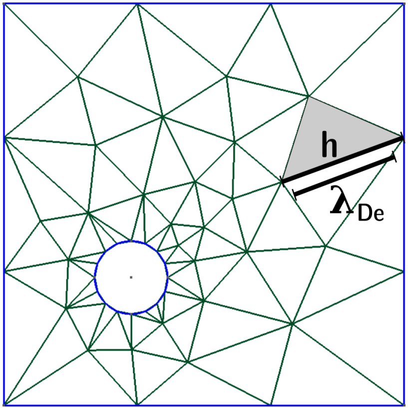


Figure: Illustration of cell diameter

Birdsall and Langdon, *Plasma Physics via Computer Simulation*
Hockney and Eastwood, *Computer Simulation Using Particles*
Marholm, PhD thesis

Outline

Why supercomputers?

Larger simulations:
larger geometries, more particles

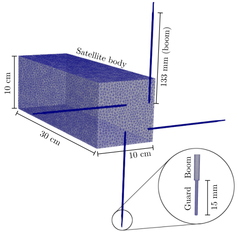


Figure: Large PTetra simulations of a CubeSat

Marholm, Marchand, *et al.*, DOI: 10.1109/TPS.2019.2915810

12 simulations $\times$ 16 cores $\times$ 8 weeks
$=$ 1536 core weeks (rough numbers)

Why supercomputers?

Larger simulations:
larger geometries, more particles

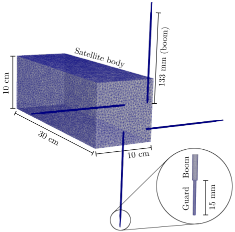


Figure: Large PTetra simulations of a CubeSat
Marholm, Marchand, *et al.*, DOI: 10.1109/TPS.2019.2915810

12 simulations × 16 cores × 8 weeks
= 1536 core weeks (rough numbers)

More simulations:
embarrasingly parallel sweeps of parameters

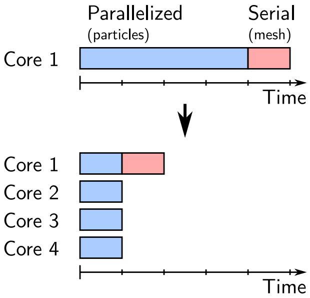
$\frac{e\Phi_p}{kT_e}$	r_p/λ_{De}				
	1.0	2.0	3.0	5.0	10.0
0	0.974	0.962	0.956	0.957	0.940
1	1.553	1.543	1.538	1.545	1.483
2	1.945	1.937	1.920	1.895	1.788
3	2.269	2.257	2.231	2.176	1.957
5	2.800	2.717	2.729	2.566	2.226
10	3.764	3.682	3.581	3.254	2.661
15	4.562	4.374	4.207	3.807	2.922
20	5.163	4.978	4.831	4.251	3.132
25	5.688	5.492	5.207	4.520	3.325

Figure: 45 of 594 small PUNC++ simulations of probes Darian, Marholm, *et al.*, DOI: 10.1088/1361-6587/ab27ff

594 simulations × 1 core × 1 week
= 594 core weeks (rough numbers)

Outline

Parallel programming



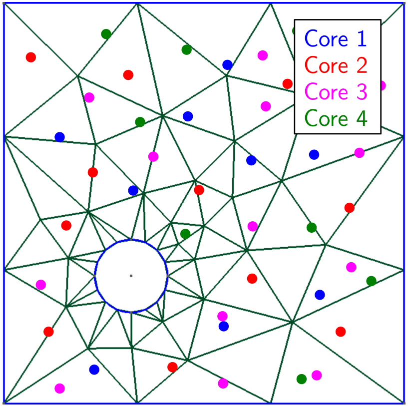


Figure: Different particles handled by different cores

Example of parallelized mesh part

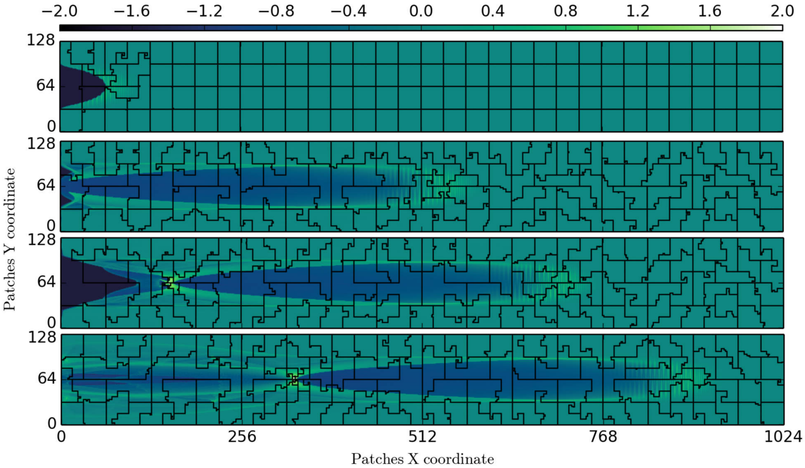
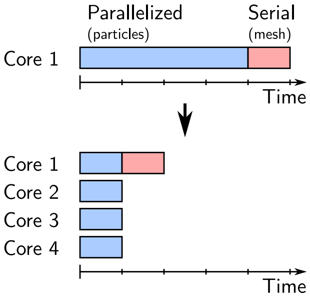


Figure: Load balancing in SmileiPIC https://smileipic.github.io/Smilei/highlights.html

Parallel programming



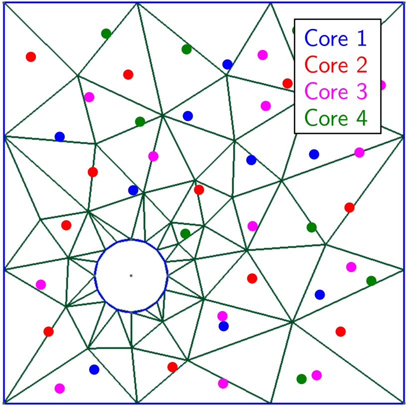
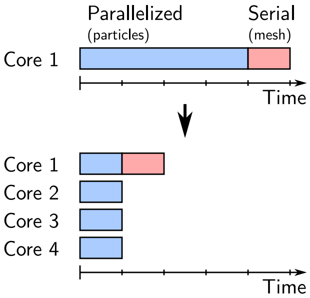


Figure: Different particles handled by different cores

Amdahl's law



$$\text{Speedup} = \frac{N}{(1-p)N + p}$$

p – Parallelized fraction

N – Number of processes

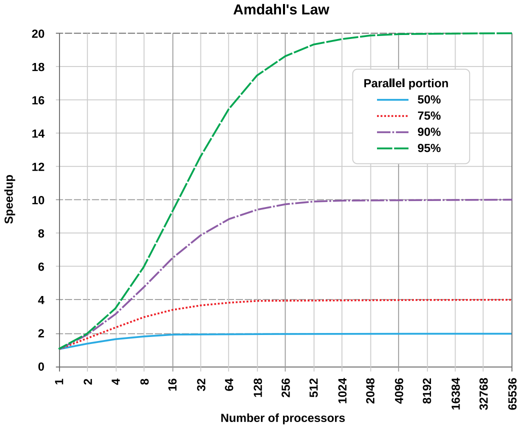
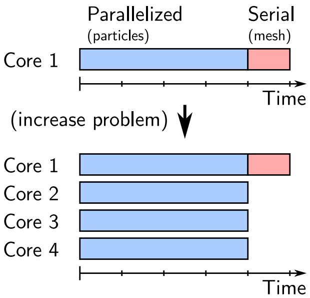


Figure: From Wikipedia

Gustafson's law



$$\text{Speedup} = (1-p) + pN$$

p – Parallelized fraction

N – Number of processes

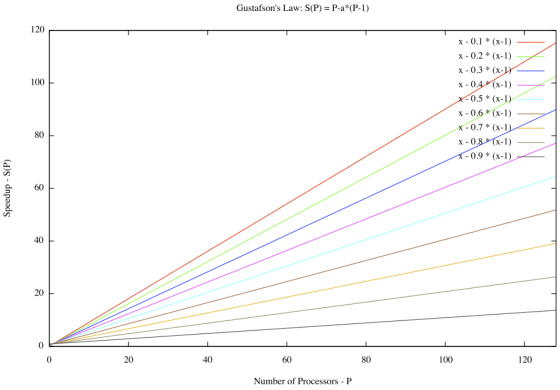


Figure: From Wikipedia

Strong and weak scaling

Strong scaling:

- Fixed problem size (Amdahl)
- Faster with more cores

Weak scaling:

- Fixed size per core (Gustafson)
- Similar execution time with more cores

Example: OSIRIS – Cartesian PIC code
10^{10} cells, 10^{13} particles, ~ 2 PFlop/s

Fonseca *et al.*, PPCF 55 (2013)

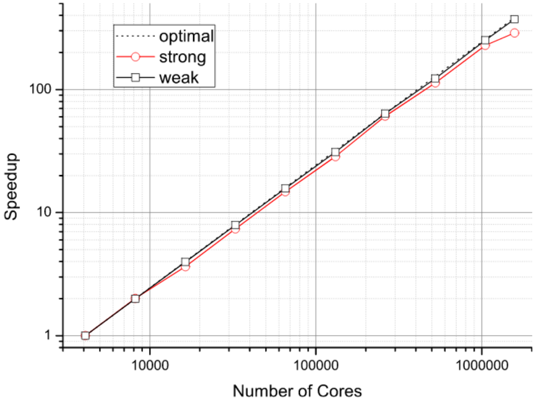
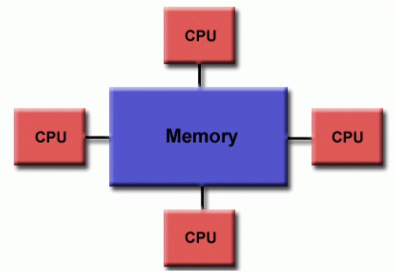


Figure: Scaling of OSIRIS (2013)

Memory architectures

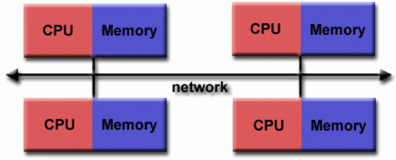
Shared memory (multicore PC, GPU):



Every core can read/write to all the memory. Must synchronize.

- OpenMP for CPUs
- OpenCL/CUDA for GPUs

Distributed memory:



Every core has it's own memory, and the cores must send each other data.

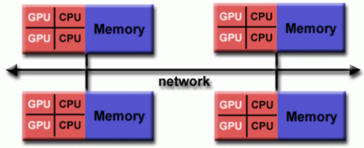
- Message Passing Interface (MPI)

https://hpc.llnl.gov/documentation/tutorials/introduction-parallel-computing-tutorial

Outline

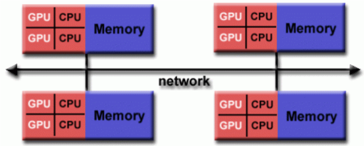
Supercomputers

Supercomputer =
many computers (nodes) + fast network



Supercomputers

Supercomputer =
many computers (nodes) + fast network



Typical storage of a particle:

x	y	z	v_x	v_y	v_z	q	m

x	y	z	v_x	v_y	v_z	w

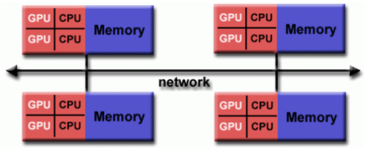
7 doubles $\times$ 8 bytes = 56 bytes

Standard node at Saga:

192 GB RAM, 40 cores
$\Rightarrow$ 4.8 GB $\approx$ 4 GB RAM/core
$\Rightarrow$ upper limit: 75 mill. particles/core (all species)
minus storage required for mesh solver

Supercomputers

Supercomputer =
many computers (nodes) + fast network



Typical storage of a particle:

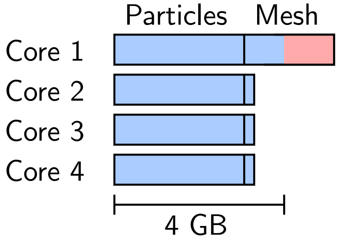
x	y	z	v_x	v_y	v_z	q	m

x	y	z	v_x	v_y	v_z	w

7 doubles $\times$ 8 bytes = 56 bytes

Standard node at Saga:

192 GB RAM, 40 cores
$\Rightarrow$ 4.8 GB $\approx$ 4 GB RAM/core
$\Rightarrow$ upper limit: 75 mill. particles/core (all species)
minus storage required for mesh solver



Usually okay if cores on same node

Supercomputers

	My laptop	Saga	Fugaku
CPU cores	2		
RAM [GB]	32		
GPUs	0		
Nodes	1		
TFlop/s	∼0.05		

Top500 List November 2021
https://www.r-ccs.riken.jp/en/fugaku
https://documentation.sigma2.no/hpc_machines/saga.html

Supercomputers

	My laptop	Saga	Fugaku
CPU cores	2	16 064	
RAM [GB]	32	99 840	
GPUs	0	32	
Nodes	1	364	
TFlop/s	∼0.05	795	

Top500 List November 2021
https://www.r-ccs.riken.jp/en/fugaku
https://documentation.sigma2.no/hpc_machines/saga.html

Supercomputers

	My laptop	Saga	Fugaku
CPU cores	2	16 064	7 630 848
RAM [GB]	32	99 840	5 087 232
GPUs	0	32	0*
Nodes	1	364	158 976
TFlop/s	∼0.05	795	537 212

*somewhat unusual. Summit has 27 648.

Top500 List November 2021
https://www.r-ccs.riken.jp/en/fugaku
https://documentation.sigma2.no/hpc_machines/saga.html

Outline

Logging in with SSH

Basic login with SSH

Example

We use Saga as example (requires user). Similar elsewhere.
HPC facilities usually have good docs. Read them.

Authenticating with keys instead of password

Create a private/public key-pair

Make sure the private key () has permissions . If not, run .

Careful! "No passphrase" is convenient, but then you must make sure private key can not be stolen (do not store on servers, etc.). See also .

Authenticating with keys instead of password

Remote machine you're logging into need the public key.
Copy contents of to a new line in on remote.

One-liner that copies public key to remote

Login with key

can be omitted if key has default name ()

Create alias for convenience

Add the following lines to to create alias

Login with alias

Similar for GitHub, etc.:
Upload public key on `github.com` → Settings → SSH and GPG keys → Add SSH key
Make sure you're connecting to GitHub via SSH and not HTTP.

Outline

Transferring files with rsync

Rsync synopsis

Example: transfer Geometry folder to MyProjects folder on Saga

Example: transfer it back

Unlike SCP, rsync continues on interrupted files next time.

Transferring files via mounted directory

Mount a remote directory to a local one

Use like any other folder.

Unmount when done

Outline

Job scripts (Slurm)

(inside simulation folder)

Job scripts (Slurm)

To allocate entire nodes

Running on multiple cores on a workstation

Outline

Managing jobs (Slurm)

Submit job

Inspect queue

Cancel job

For other workload managers: `https://slurm.schedmd.com/rosetta.pdf`

Managing jobs

Inspect quotas (Sigma2-specific)

Inspect PTetra output

Outline