Multi-GPU Acceleration of the iPIC3D Implicit Particle-in-Cell Code

Chaitanya Prasad Sishtla, Steven W. D. Chien, Vyacheslav Olshevsky, Erwin Laure, Stefano Markidis
{sishtla,wdchien,slavik,erwinl,markidis}@kth.se

Introduction

iPIC3D is a widely used massively parallel Particle-in-Cell code for the simulation of space plasmas. However, its current implementation does not support execution on multiple GPUs. In this paper, we describe the porting of iPIC3D particle mover to GPUs and the optimization steps to increase the performance and parallel scaling on multiple GPUs. We analyze the strong scaling of the mover on two GPU clusters and evaluate its performance and acceleration. The optimized GPU version which uses pinned memory and asynchronous data prefetching outperform their corresponding CPU versions by $5-10\times$ on two different systems (Tegner and Kebnekaise) equipped with NVIDIA K80 and V100 GPUs. Compared to previous works which use a simple formulation of the PIC algorithm, we port a semi-implicit PIC method and benchmark the code using a practical space plasma simulation.

Conclusions

1. We port the iPIC3D particle mover to GPUs as profiling reveals it to be the most expensive step.

2. We find that memory pinning and prefetching are essential to improve performance.

3. GPU movers outperform the purely CPU implementation by $5-10$ MPA/s.

4. The prefetch mover reaches 73% parallel efficiency on 16 K80 GPUs as compared to 44% for the naive implementation.

5. The performance of the Prefetch mover does not substantially depend on the number of particles in the simulation.

Future Work

1. Investigate the dependence of the performance of a GPU-ported particle mover on other system parameters, such as the number of predictor-corrector iterations.

2. Porting of the particle distribution function moment calculation to GPUs, or its possible merging with the mover phase of the computational cycle.

References

[1] Stefano Markidis et al. Multi-scale simulations of plasma with iPIC3D. *Mathematics and Computers in Simulation*, 80(7):1509–1519, 2010.

[2] Joachim Birn and Michael Hesse. Geospace Environment Modeling (GEM) magnetic reconnection challenge: Resistive tearing, anisotropic pressure and Hall effects. *Journal of Geophysical Research: Space Physics*, 106(A3):3737–3750, 2001.

Acknowledgements

This work has received funding from the European Commission H2020 program, Grant Agreement No. 801039 (EPiGRAM-HS). Experiments were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at PDC Center for High Performance Computing and HPC2N.

Profiling Information

Table 1: Percentage of execution time per cycle for the three PIC steps in a typical iPIC3D benchmark run on a CPU.

Part of Code	% Time Spent				
	27ppc	64ppc	125ppc	216ppc	343ppc
Fields solver	6.12	2.79	1.47	0.87	0.55
Particle mover	68.81	71.42	72.23	72.77	73.14
Moments calculation	25.04	25.76	26.31	26.35	26.29

Porting to Multi-GPU Systems

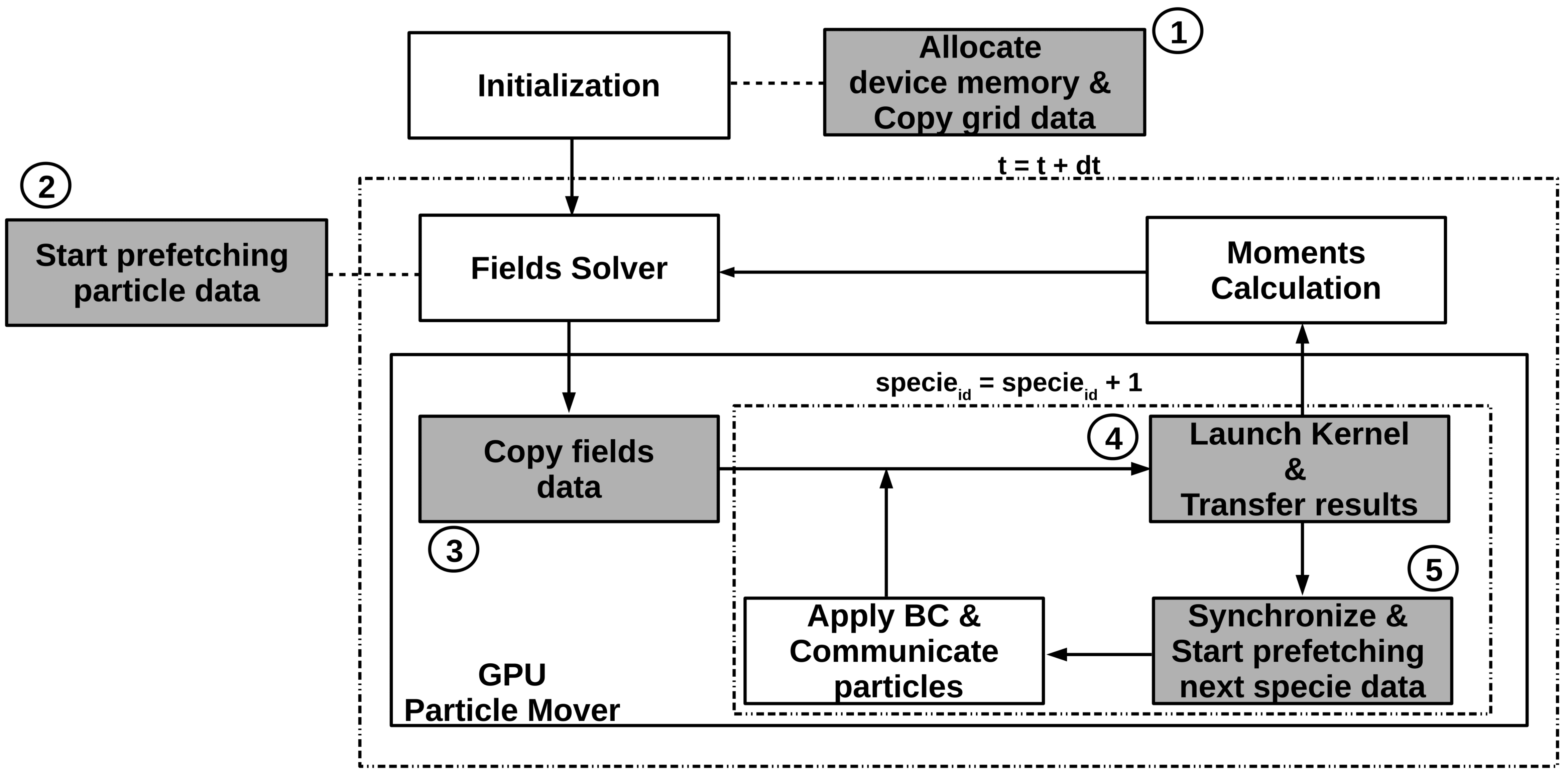
Initialization
Allocate device memory & Copy grid data
1
t = t + dt
2
Start prefetching particle data
Fields Solver
Moments Calculation
$\text{specie}_{id} = \text{specie}_{id} + 1$
Copy fields data
3
4
Launch Kernel & Transfer results
5
Apply BC & Communicate particles
Synchronize & Start prefetching next specie data
GPU Particle Mover

Figure 1: The flowchart of the iPIC3D code with the GPU particle mover using data prefetching is shown. The white blocks correspond to instructions executed on the CPU while the grey blocks correspond to CUDA code. Dashed lines indicate where in the host the relevant CUDA code is called.

Results

Table 2: The average time spent in the particle mover over 10 cycles using a single GPU (one MPI process) in different testing environments.

Type of Node	Particle mover execution times (in seconds)			
	CPU	Naive	Pinned	Prefetch
Tegner (Haswell+K80)	15.33	3.28	3.05	2.44
Kebnekaise (Broadwell+K80)	15.20	3.84	3.44	2.87
Kebnekaise (Skylake+V100)	36.82	4.20	2.02	1.43

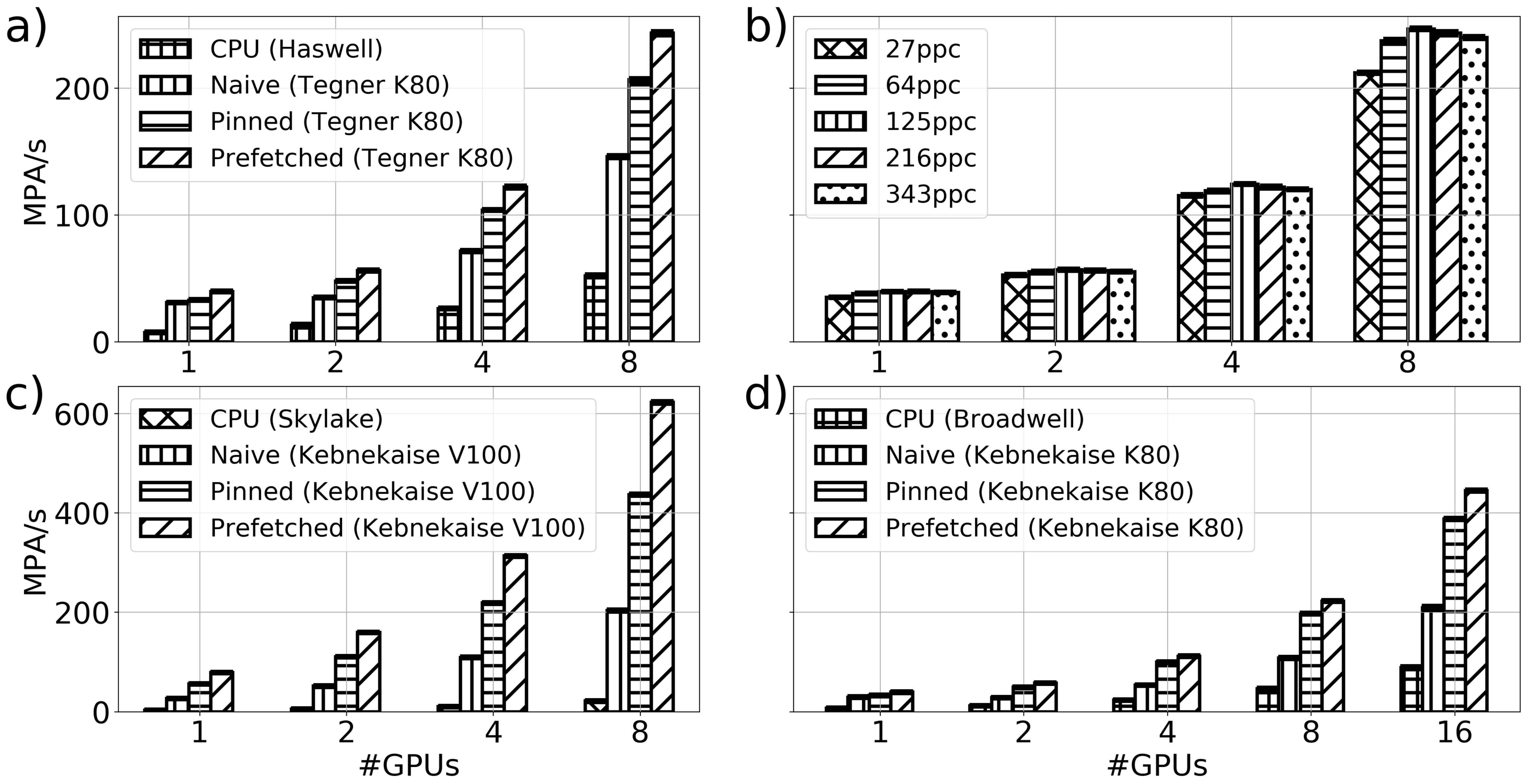

Figure 2: a) The performance of the GPU porting schemes compared on Tegner (using the Haswell CPU and the K80 node). b) The performance of the prefetched GPU porting schemes compared on the K80 nodes of Tegner by varying the number of particles per cell in the simulation. The performance of the GPU porting schemes compared for Kebnekaise. c) nodes with Broadwell CPUs and K80 GPUs; d) nodes with Skylake CPUs and V100 GPUs.