PPP – projekt 1

Simulácia ohrevu vežovitého chladiča

Matej Karas – xkaras34

Cieľom projektu bola implementácia simulácie ohrevu vežovitého chladiča procesoru za použitia MPI určenú pre výpočetný cluster.

Aplikácia bola testovaná pod OpenMPI verzie 4.0.3 a Intel MPI 2018a. Verzia 2020a obsahuje nejaký bug, pri ktorom zasielanie halo zón nefungovalo (konkrétne nefungovali offsety).

Taktiež pri použití agresívnych optimalizácií za použitia prekladača Intel, vzniká chyba floating point operácií, ktorá sa pri dostatočnom počte iterácií akumuluje a presiahne dovolenú odchíľku (zlyhanie testu voči sekvenčnej verzii). Pri použití OpenMPI s GCC tento problém nikdy nenastal a diferencia bola vždy nulová.

Dosiahnuté výsledky

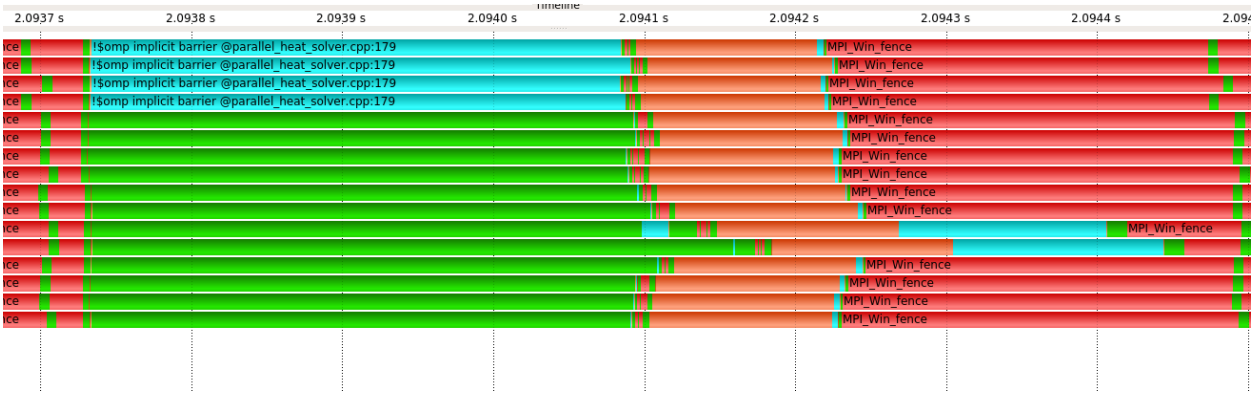
Varianta hybridné MPI (na jeden MPI proces je priradených 6 vláken) s 2D dekompozíciou sa preukázala byť najefektívnejšia. Je to dané tým, že nedochádza k tak veľkej komunikácii, ako keby sa použilo také isté množstvo výpočtových jadier, avšak s alokáciou jeden proces na jedno jadro.

Pri všetkých variantách algoritmus dobre silne škáluje, pokiaľ je úloha výpočtovo náročná a teda dochádza k dostatočnému prekrytiu komunikácie a teda približne pri maticiach 2048x2048 a viac.

Čo sa týka slabého škálovaniam, grafy naznačujú, že pokiaľ sa lokálna matica zmestí do pamäti cache, je algoritmus efektívny, a niekedy dochádza až k super-škálovaniu – zdvojnásobením počtu výpočtových jadier a veľkosti úlohy, dôjde k zrýchleniu.

Dekompozícia 1D a 2D škáluje takmer rovnako, avšak 2D dekompozícia je efektívnejšia. Je to dané tým, že pri 2D dekompozícii dochádza k menšej komunikácii.

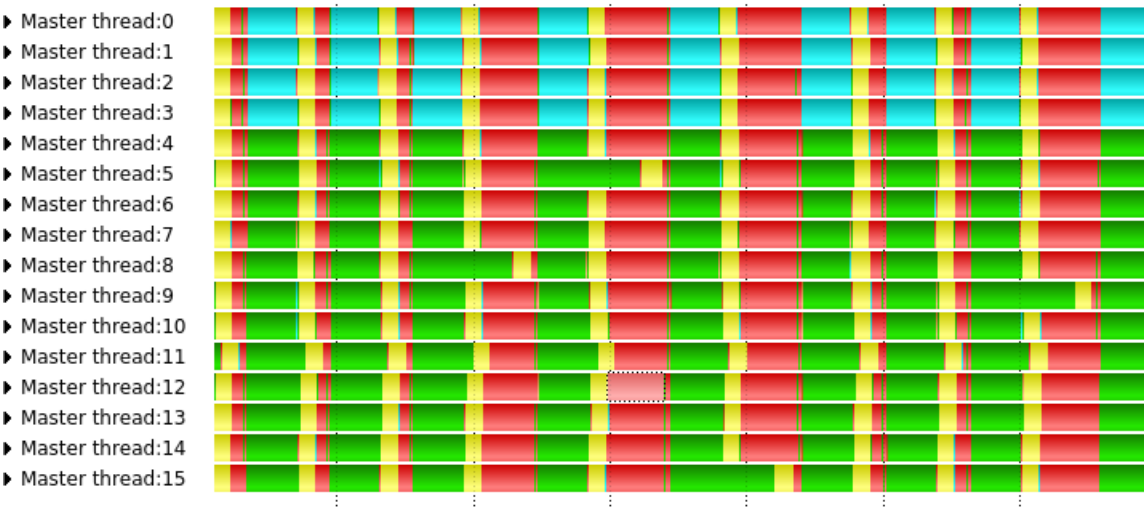
Použitie paralelného I/O sa vyplácalo len pokiaľ lokálna matica bola dostatočne veľká – v tom príde bolo možne pozorovať až takmer dvoj-násobné zrýchlenie oproti zápisu jedným procesom. V opačnom prípade, pokiaľ bola lokálna matica malá (4x4 alebo 8x8) bol zápis až dvojnásobne pomalší.



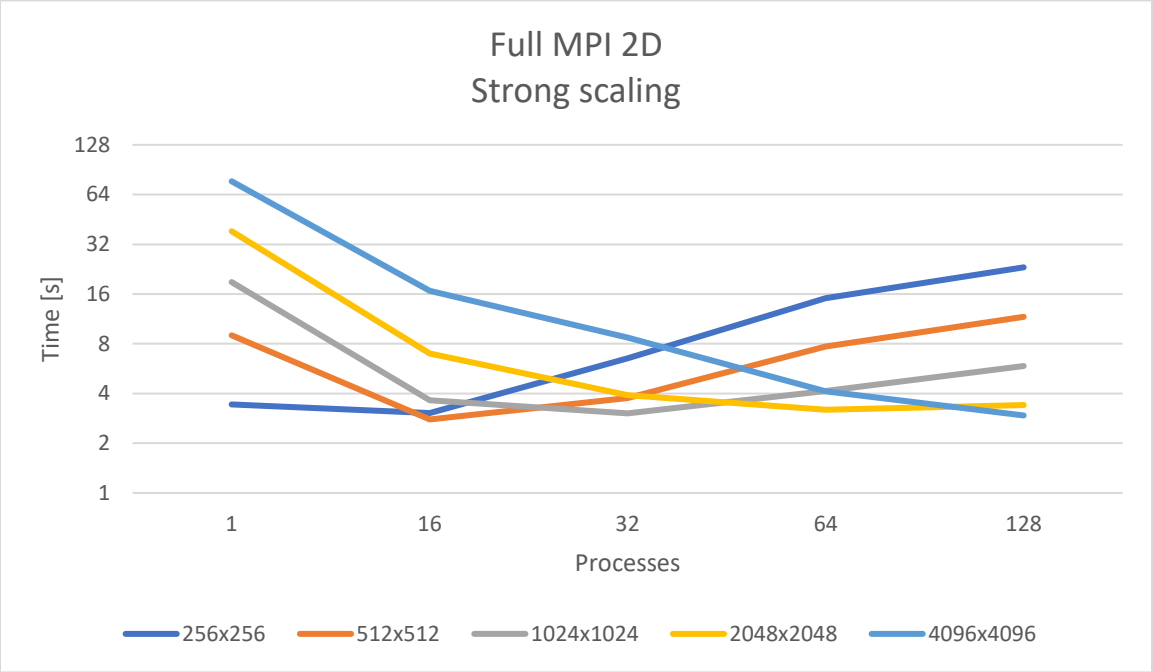
Na obrázku je možné vidieť jednu iteráciu výpočtu. Najskôr sa pomocou MPI_*win_fence()* zosynchronizuje výpočet tak, aby potrebné dáta z predošlej iterácie boli už zapísané v pamäti. Nasleduje výpočet halo-zón (zelená oblasť v obrázku nižšie). Nepoužitím vektorových inštrukcií a zároveň značným vyplachovaním pamätí cache, je paradoxne výpočet týchto zón pomalší ako samotný výpočet zbytku matice (oranžová oblasť).

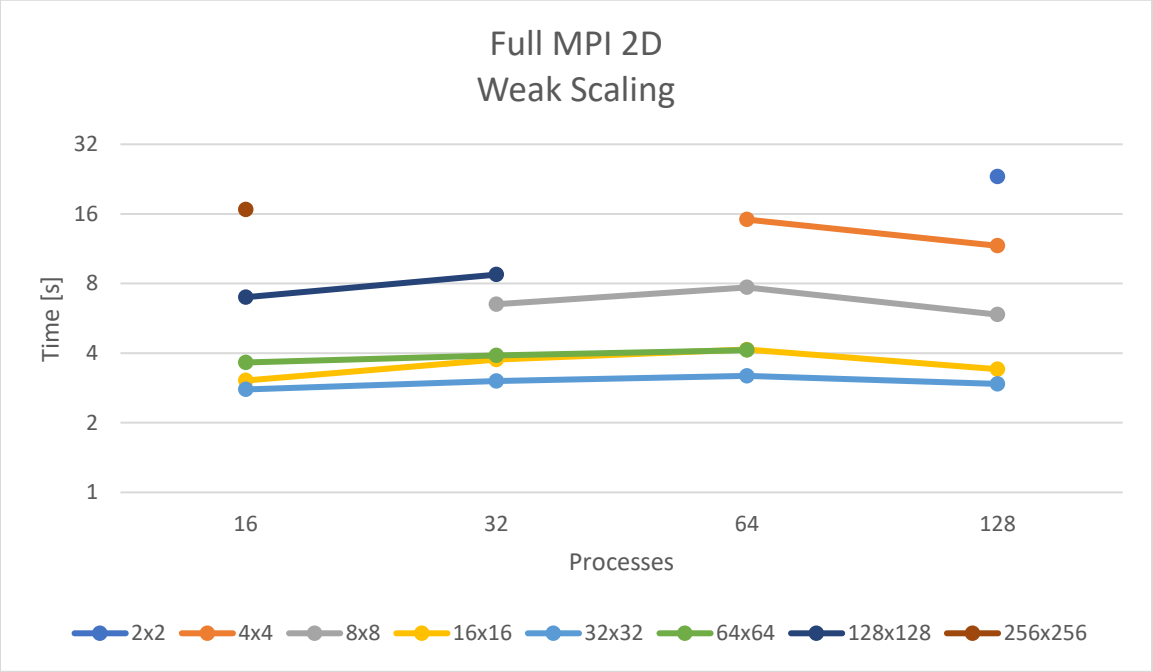
Vďaka prekrytiu komunikácie zasielania halo-zón výpočtom, nemusí byť umiestnené ďalšie synchronizačné primitívum a teda viacej času strávi algoritmus počítaním, než čakaním.

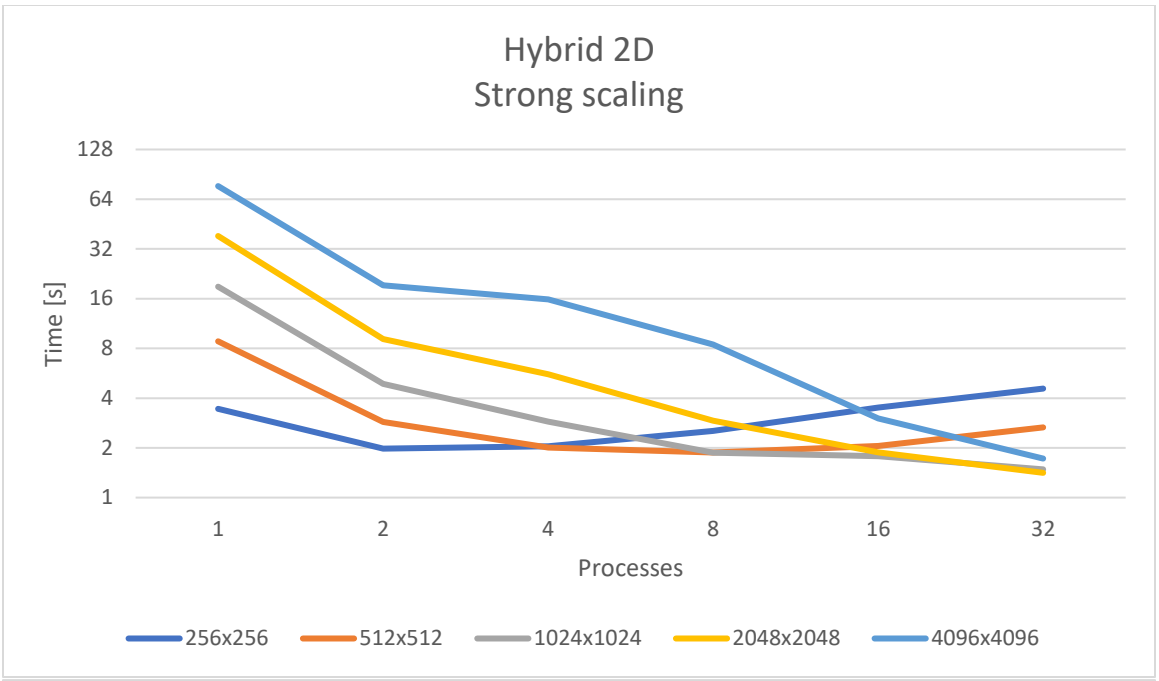
V niektorých iteráciách sa stalo, že pri zakladaní OMP vláken, resp. paralelného úseku, sa založenie niektorého vlákna veľmi oneskorilo, a teda všetky ostatné procesy museli čakať, kým vypočíta svoju pridelenú prácu. V obrázku nižšie sa to premietlo do červených oblastí, resp. čakanie na *MPI_Win_fence().* V opačnom prípade je synchronizácia relatívne zanedbateľná voči výpočtu.

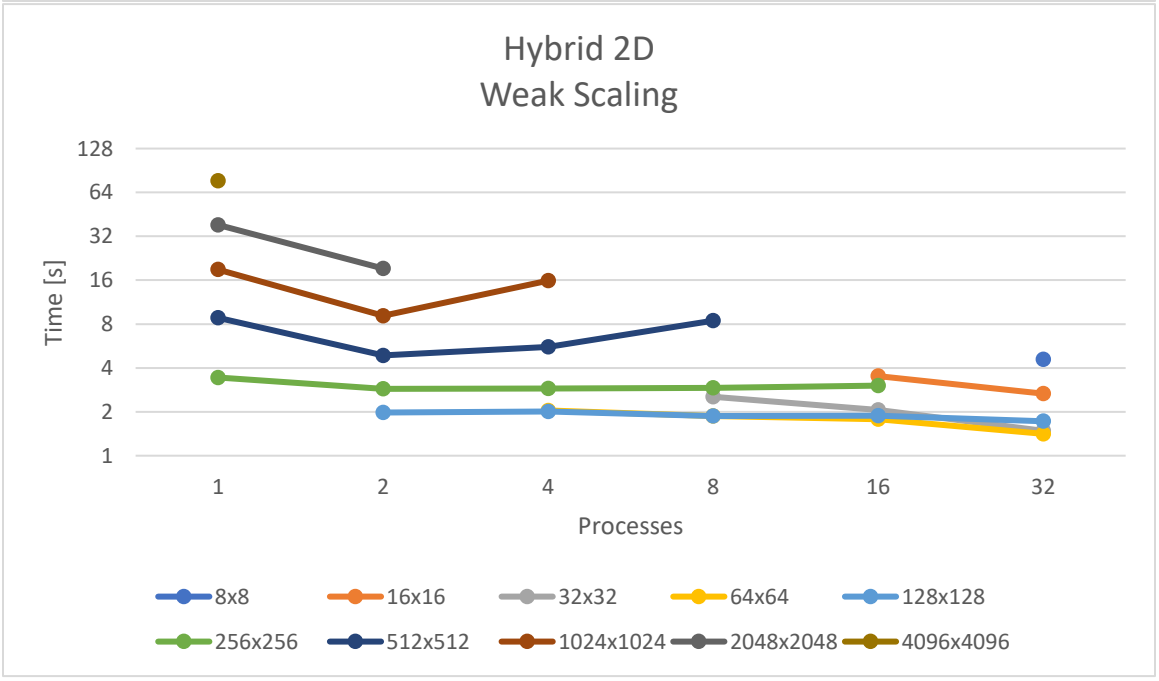


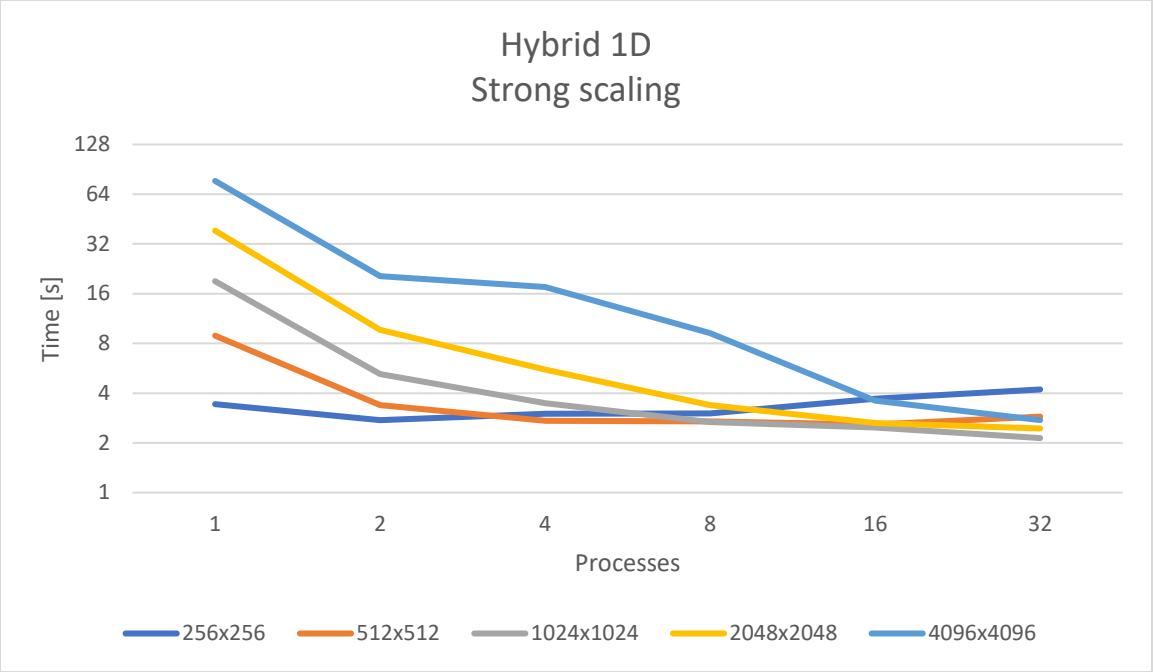
Grafy

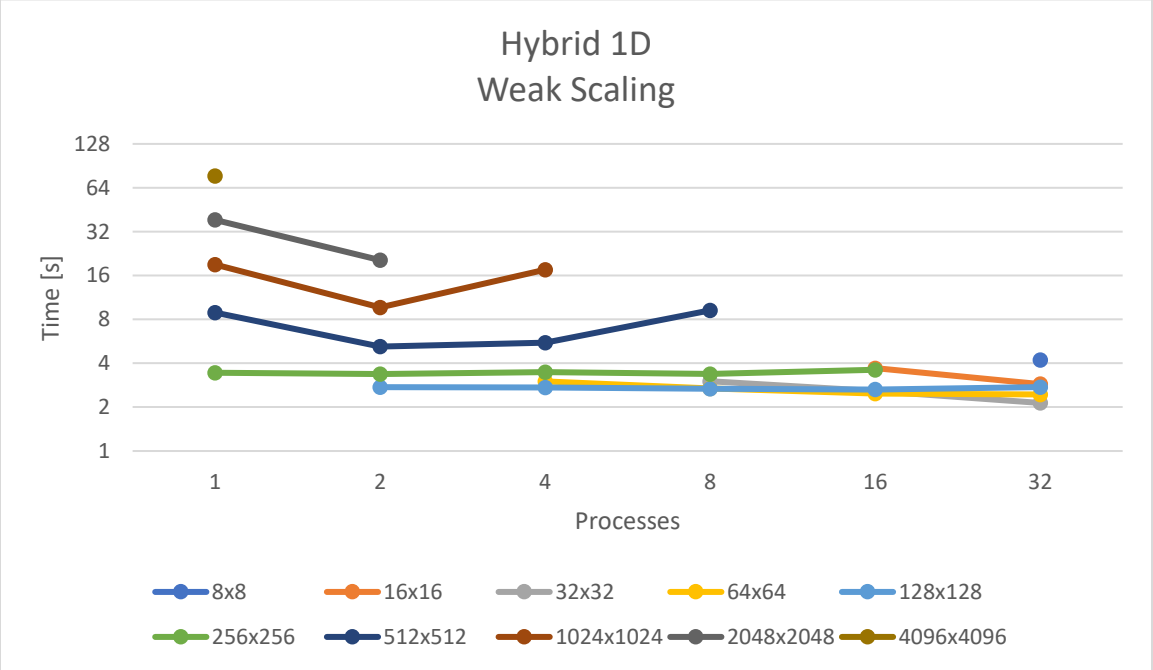

Full MPI 2D
Strong scaling
128
64
32
16
8
4
2
1
Time [s]
1
16
32
64
128
Processes
256x256
512x512
1024x1024
2048x2048
4096x4096

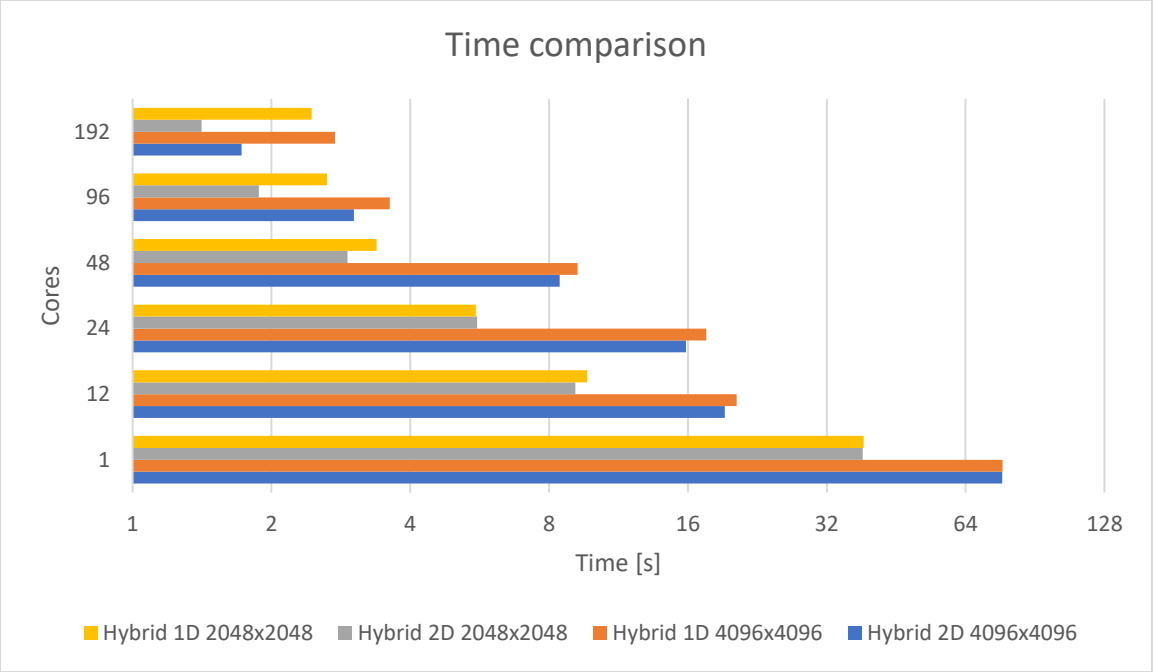

Full MPI 2D
Weak Scaling
32
16
8
4
2
1
Time [s]
16
32
64
128
Processes
2x2
4x4
8x8
16x16
32x32
64x64
128x128
256x256


Hybrid 2D
Strong scaling
Time [s]
128
64
32
16
8
4
2
1
1
2
4
8
16
32
Processes
256x256
512x512
1024x1024
2048x2048
4096x4096


Hybrid 2D
Weak Scaling
Time [s]
128
64
32
16
8
4
2
1
1
2
4
8
16
32
Processes
8x8
16x16
32x32
64x64
128x128
256x256
512x512
1024x1024
2048x2048
4096x4096


Hybrid 1D
Strong scaling
Time [s]
128
64
32
16
8
4
2
1
1
2
4
8
16
32
Processes
256x256
512x512
1024x1024
2048x2048
4096x4096


Hybrid 1D
Weak Scaling
Time [s]
128
64
32
16
8
4
2
1
1
2
4
8
16
32
Processes
8x8
16x16
32x32
64x64
128x128
256x256
512x512
1024x1024
2048x2048
4096x4096


Time comparison
Cores
192
96
48
24
12
1
1
2
4
8
16
32
64
128
Time [s]
Hybrid 1D 2048x2048
Hybrid 2D 2048x2048
Hybrid 1D 4096x4096
Hybrid 2D 4096x4096