The University of Texas at Austin

HPSF CONFERENCE 2025

May 7th, 2025

Performance Portable Fast Ewald Summation

Kokkos User Group | Chicago, IL

Gabriel Kosmacher, Ziyu Du, Joar Bagge, George Biros

ODEN INSTITUTE

PECOS Predictive Engineering & Computational Science

PSAAP III

- Consider periodic N-body sums of the form

$$\boldsymbol{u}(\boldsymbol{x}_i) = \sum_{j=1}^{N_s} \sum_{\boldsymbol{p}\in\mathcal{P}} \boldsymbol{G}(\boldsymbol{x}_i - \boldsymbol{y}_j + \boldsymbol{p})\boldsymbol{f}(\boldsymbol{y}_j) \quad i = 1, \ldots, N_t.$$

- Ewald's method splits

$$\boldsymbol{u}(\boldsymbol{x}_i) = \boldsymbol{u}^{\mathrm{N}}(\boldsymbol{x}_i) + \boldsymbol{u}^{\mathrm{F}}(\boldsymbol{x}_i)$$

  by setting $\boldsymbol{G}(\boldsymbol{r}) = \boldsymbol{G}^{\mathrm{N}}(\boldsymbol{r};\xi) + \boldsymbol{G}^{\mathrm{F}}(\boldsymbol{r};\xi)$.

- For the electrostatic potential,

$$\boldsymbol{G}(\boldsymbol{r}) = \frac{1}{\|\boldsymbol{r}\|} = \underbrace{\frac{\operatorname{erfc}(\xi\|\boldsymbol{r}\|)}{\|\boldsymbol{r}\|}}_{\boldsymbol{G}^{\mathrm{N}}} + \underbrace{\frac{\operatorname{erf}(\xi\|\boldsymbol{r}\|)}{\|\boldsymbol{r}\|}}_{\boldsymbol{G}^{\mathrm{F}}}.$$

- Consider periodic N-body sums of the form

$$\boldsymbol{u}(\boldsymbol{x}_i) = \sum_{j=1}^{N_s} \sum_{\boldsymbol{p}\in\mathcal{P}} \boldsymbol{G}(\boldsymbol{x}_i - \boldsymbol{y}_j + \boldsymbol{p})\boldsymbol{f}(\boldsymbol{y}_j) \quad i = 1, \ldots, N_t.$$

- Ewald's method splits

$$\boldsymbol{u}(\boldsymbol{x}_i) = \boldsymbol{u}^{\mathrm{N}}(\boldsymbol{x}_i) + \boldsymbol{u}^{\mathrm{F}}(\boldsymbol{x}_i)$$

by setting $\boldsymbol{G}(\boldsymbol{r}) = \boldsymbol{G}^{\mathrm{N}}(\boldsymbol{r};\xi) + \boldsymbol{G}^{\mathrm{F}}(\boldsymbol{r};\xi)$.

- For the electrostatic potential,

$$G(r) = \frac{1}{\|\boldsymbol{r}\|} = \underbrace{\frac{\operatorname{erfc}(\xi\|\boldsymbol{r}\|)}{\|\boldsymbol{r}\|}}_{G^{\mathrm{N}}} + \underbrace{\frac{\operatorname{erf}(\xi\|\boldsymbol{r}\|)}{\|\boldsymbol{r}\|}}_{G^{\mathrm{F}}}.$$

- Consider periodic N-body sums of the form

$$\boldsymbol{u}(\boldsymbol{x}_i) = \sum_{j=1}^{N_s} \sum_{\boldsymbol{p}\in\mathcal{P}} \boldsymbol{G}(\boldsymbol{x}_i - \boldsymbol{y}_j + \boldsymbol{p})\boldsymbol{f}(\boldsymbol{y}_j) \quad i = 1, \ldots, N_t.$$

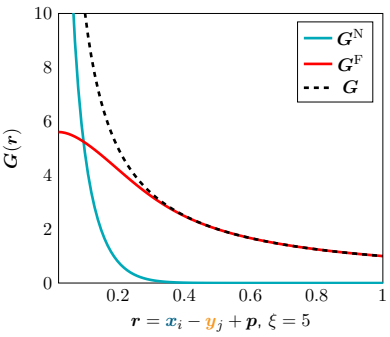
- Ewald's method splits

$$\boldsymbol{u}(\boldsymbol{x}_i) = \boldsymbol{u}^{\mathrm{N}}(\boldsymbol{x}_i) + \boldsymbol{u}^{\mathrm{F}}(\boldsymbol{x}_i)$$

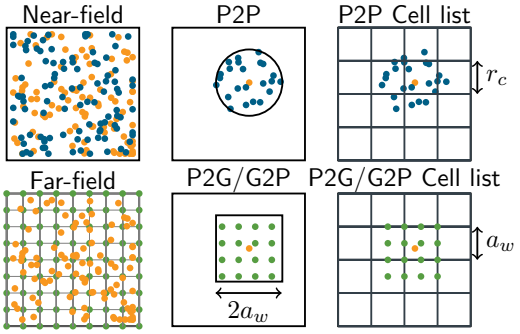
  by setting $\boldsymbol{G}(\boldsymbol{r}) = \boldsymbol{G}^{\mathrm{N}}(\boldsymbol{r};\xi) + \boldsymbol{G}^{\mathrm{F}}(\boldsymbol{r};\xi)$.

- For the electrostatic potential,

$$\boldsymbol{G}(\boldsymbol{r}) = \frac{1}{\|\boldsymbol{r}\|} = \underbrace{\frac{\operatorname{erfc}(\xi\|\boldsymbol{r}\|)}{\|\boldsymbol{r}\|}}_{\boldsymbol{G}^{\mathrm{N}}} + \underbrace{\frac{\operatorname{erf}(\xi\|\boldsymbol{r}\|)}{\|\boldsymbol{r}\|}}_{\boldsymbol{G}^{\mathrm{F}}}.$$



- *Padded cell lists* $\mathcal{C}^{\mathrm{N}}$ accelerate **P2P**, **P2G**, and **G2P** by assigning each Kokkos team T to a cell β so each thread $t_x \in T$ processes spatially close data.
- $\mathcal{C}^{\mathrm{N}}$ is constructed in two passes though an unsorted list of particles



- For the stokes potential,

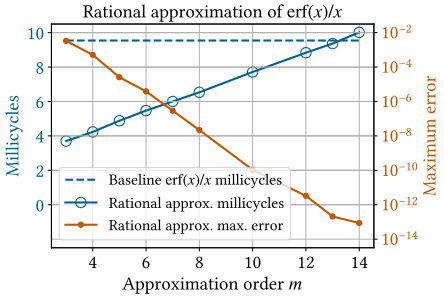
$$\boldsymbol{G}^{\mathrm{N}}(\boldsymbol{r}) = \left(\boldsymbol{I} + \frac{\boldsymbol{r} \otimes \boldsymbol{r}}{\|\boldsymbol{r}\|^2}\right)\left(\operatorname{erfc}(\xi\|\boldsymbol{r}\|) + \frac{\|\boldsymbol{r}\| 2\xi \mathrm{e}^{-\xi^2\|\boldsymbol{r}\|^2}}{\sqrt{\pi}}\right) - \boldsymbol{I}\frac{4\xi \mathrm{e}^{-\xi^2\|\boldsymbol{r}\|^2}}{\sqrt{\pi}}.$$

Table: FP64 functions on the NVIDIA H200 GPU, measured with NVIDIA Nsight Compute.

Operation	Millicycles	Flop-equivalents	Actual flops
DFMA	0.121	2	2
DADD	0.127	2.10	1
DMUL	0.127	2.10	1
DIV	2.11	35.0	15 (2)
RSQRT	1.82	30.2	8
EXP	3.07	50.8	30
ERFC	16.1	267	112

- For the stokes potential,

$$\boldsymbol{G}^{\mathrm{N}}(\boldsymbol{r}) = \left(\boldsymbol{I} + \frac{\boldsymbol{r} \otimes \boldsymbol{r}}{\|\boldsymbol{r}\|^2}\right)\left(\operatorname{erfc}(\xi\|\boldsymbol{r}\|) + \frac{\|\boldsymbol{r}\| 2\xi \mathrm{e}^{-\xi^2\|\boldsymbol{r}\|^2}}{\sqrt{\pi}}\right) - \boldsymbol{I}\frac{4\xi \mathrm{e}^{-\xi^2\|\boldsymbol{r}\|^2}}{\sqrt{\pi}}.$$



Algorithm Design

The window $w(\boldsymbol{r}) = \sum_{i=1}^{d} w_0(r_i)$ is identical in each dimension. How can we use this structure to build an efficient parallel scheme?

`P2G-BASE`: Too many atomics!

`P2G-GRID`: Redundant work!

`P2G-HYBRID`: Best of both worlds?

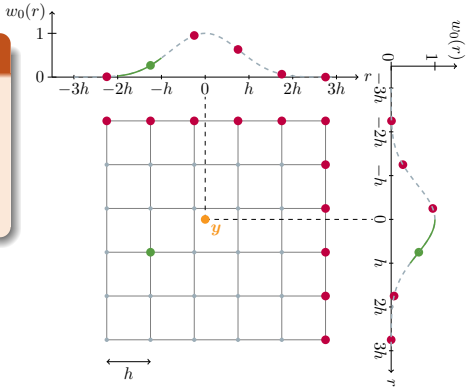
Algorithm Design

The window $w(\boldsymbol{r}) = \sum_{i=1}^{d} w_0(r_i)$ is identical in each dimension. How can we use this structure to build an efficient parallel scheme?

`P2G-BASE`: Too many atomics!

`P2G-GRID`: Redundant work!

`P2G-HYBRID`: Best of both worlds?



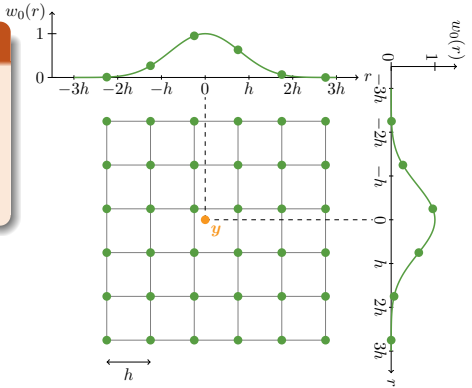
Algorithm Design

The window $w(\boldsymbol{r}) = \sum_{i=1}^{d} w_0(r_i)$ is identical in each dimension. How can we use this structure to build an efficient parallel scheme?

P2G-BASE: Too many atomics!

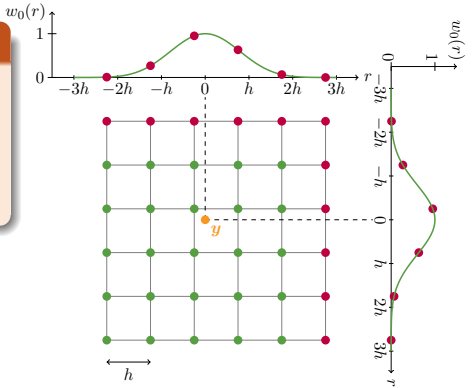
P2G-GRID: Redundant work!

P2G-HYBRID: Best of both worlds?



Algorithm Design

The window $w(\boldsymbol{r}) = \sum_{i=1}^{d} w_0(r_i)$ is identical in each dimension. How can we use this structure to build an efficient parallel scheme?



P2G-BASE: Too many atomics!

P2G-GRID: Redundant work!

P2G-HYBRID: Best of both worlds?

Results: Performance Model

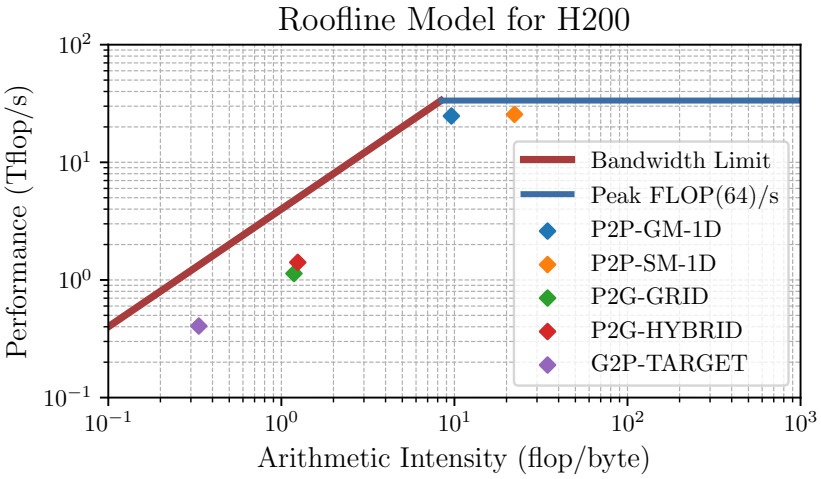


Table: P2P results on an NVIDIA H200 GPU.
Higher $\frac{N_s}{\mu s}$ is better.

N_t	method	$s = 256$		
		(b_x, b_y)	$\frac{N_t}{\mu s}$	Flops
10^6	GM-1D	$(64, 1)$	14.8	71%
	GM-2D	$(1, 32)$	11.8	56%
	SM-1D	$(32, 1)$	15.2	72%
	SM-2D	$(64, 2)$	12.4	59%
$4 \cdot 10^6$	GM-1D	$(64, 1)$	15.3	73%
	GM-2D	$(8, 32)$	11.9	57%
	SM-1D	$(32, 1)$	15.5	74%
	SM-2D	$(64, 2)$	12.4	59%

Table: P2G results on an NVIDIA H200 GPU.
Higher $\frac{N_s}{\mu s}$ is better.

N_s	method	$P = 8, E = 10^{-9}$		
		$\frac{N_s}{\mu s}$	Spd	Mops
10^6	BASE	10.1	—	22%
	GRID	11.9	1.2×	26%
	HYBRID	63.8	6.3×	30%
$4 \cdot 10^6$	BASE	8.5	—	18%
	GRID	11.1	1.3×	24%
	HYBRID	59.9	7.1×	28%

Table: P2P results on an NVIDIA H200 GPU. Higher $\frac{N_s}{\mu s}$ is better.

N_t	method	$s = 256$		
		(b_x, b_y)	$\frac{N_t}{\mu s}$	Flops
10^6	GM-1D	$(64, 1)$	14.8	71%
	GM-2D	$(1, 32)$	11.8	56%
	SM-1D	$(32, 1)$	15.2	72%
	SM-2D	$(64, 2)$	12.4	59%
$4 \cdot 10^6$	GM-1D	$(64, 1)$	15.3	73%
	GM-2D	$(8, 32)$	11.9	57%
	SM-1D	$(32, 1)$	15.5	74%
	SM-2D	$(64, 2)$	12.4	59%

Table: P2G results on an NVIDIA H200 GPU. Higher $\frac{N_s}{\mu s}$ is better.

N_s	method	$P = 8, E = 10^{-9}$		
		$\frac{N_s}{\mu s}$	Spd	Mops
10^6	BASE	10.1	—	22%
	GRID	11.9	1.2×	26%
	HYBRID	63.8	6.3×	30%
$4 \cdot 10^6$	BASE	8.5	—	18%
	GRID	11.1	1.3×	24%
	HYBRID	59.9	7.1×	28%

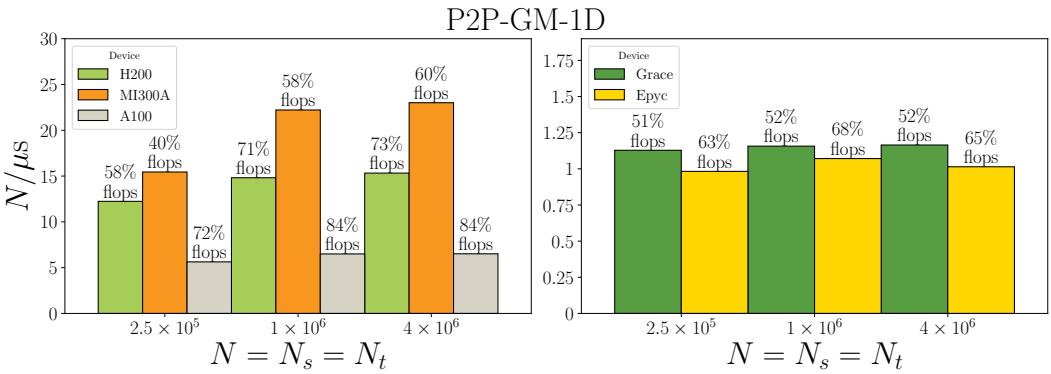


Figure: Particle-to-particle performance portability (higher is better), $s = 256$.

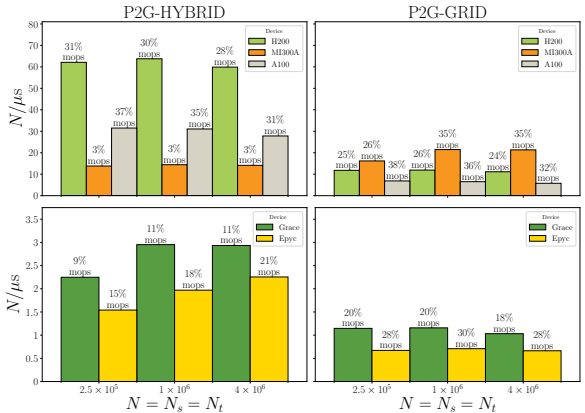


Figure: Particle-to-grid performance portability (higher is better). $E = 10^{-9}$, $s = 224$.

Conclusion

- Introduced algorithms for spectral Stokes Ewald sum with performance analysis.
- Library implemented in PyKokkos, MPI results show preliminary overheads.
- Limitations (i.e., future work):
  - No single-precision tests.
  - CPU kernels and MPI communication not optimized.
  - Nonuniformity effects not studied.
  - Fix shared memory slowdowns on AMD GPUs.
  - No multi-GPU NUMA scaling.

Thank You!

Gabriel Kosmacher
gkosmacher@utexas.edu

Conclusion

- Introduced algorithms for spectral Stokes Ewald sum with performance analysis.
- Library implemented in PyKokkos, MPI results show preliminary overheads.
- Limitations (i.e., future work):
  - No single-precision tests.
  - CPU kernels and MPI communication not optimized.
  - Nonuniformity effects not studied.
  - Fix shared memory slowdowns on AMD GPUs.
  - No multi-GPU NUMA scaling.

Thank You!

Gabriel Kosmacher
gkosmacher@utexas.edu

- **Particle-to-particle (P2P):**

$$\boldsymbol{u}^{\mathrm{N}}(\boldsymbol{x}_i) := \sum\nolimits_{j\in[N_s]} \sum\nolimits_{\boldsymbol{p}\in\mathcal{P}} \boldsymbol{G}^{N}(\boldsymbol{x}_i - \boldsymbol{y}_j + \boldsymbol{p})\boldsymbol{f}(\boldsymbol{y}_j) \qquad \mathcal{O}(27sN_t)$$

- **Particle-to-grid (P2G):**

$$\boldsymbol{\phi}_h(\boldsymbol{g}_\ell) := \sum\nolimits_{j\in[N_s]} \sum\nolimits_{\boldsymbol{p}\in\mathcal{P}} w(\boldsymbol{g}_\ell - \boldsymbol{y}_j + \boldsymbol{p})\boldsymbol{f}(\boldsymbol{y}_j) \qquad \mathcal{O}(N_s P^3)$$

- **Fourier Grid Convolution (FGC):**

$$\boldsymbol{v}_h(\boldsymbol{g}_\ell) := \mathtt{IFFT}\left\{\widehat{\boldsymbol{G}}^{\mathrm{F}}(\boldsymbol{\kappa}_\ell)[\widehat{w}(\boldsymbol{\kappa}_\ell)]^{-2}\mathtt{FFT}\{\boldsymbol{\phi}_h(\boldsymbol{g}_\ell)\}\right\} \qquad \mathcal{O}(N_g \log N_g)$$

- **Grid-to-particle (G2P):**

$$\boldsymbol{u}_h^{\mathrm{F}}(\boldsymbol{x}_i) := h^3 \sum\nolimits_{\ell\in[N_g]} \sum\nolimits_{\boldsymbol{p}\in\mathcal{P}} w(\boldsymbol{x}_i - \boldsymbol{g}_\ell + \boldsymbol{p})\boldsymbol{v}_h(\boldsymbol{g}_\ell) \qquad \mathcal{O}(N_t P^3)$$

0: **parfor** $\boldsymbol{x}_i \in \beta$ **do** // Team gets β, TeamThreadRange over $\boldsymbol{x}_i$
1: **parfor** $c \leftarrow 1, \ldots, \lceil s/|\alpha_c| \rceil$ **do** // VectorThreadRange over $\boldsymbol{y}_j$
2: $\forall \boldsymbol{y}_j \in \alpha_c, \textsc{LoadScratch}(\boldsymbol{y}_j, \boldsymbol{f}(\boldsymbol{y}_j))$ // GM/SM
3: $\_\_\textsc{TeamBarrier}(\,)$
4: **for** $\boldsymbol{y}_j \in \alpha_c$ **do**
5: $\boldsymbol{u}_\mathrm{c} \leftarrow \boldsymbol{u}_\mathrm{c} + \boldsymbol{G}^N(\boldsymbol{x}_i - \boldsymbol{y}_j + \boldsymbol{p})\boldsymbol{f}(\boldsymbol{y}_j)$ // $\boldsymbol{G}^\mathrm{N}$ is flop intensive!
6: **end for**
7: $\_\_\textsc{TeamBarrier}(\,)$
8: **end parfor**
9: $\boldsymbol{u}_h^N(\boldsymbol{x}_i) \leftarrow \textsc{VectorReduce}(\boldsymbol{u}_\mathrm{c})$ // Kokkos provided reduction
10: **end parfor**

0: **parfor** $\boldsymbol{y}_j \in \beta$ **do** // Team gets β, TeamThreadRange over $\boldsymbol{y}_j$
1: Compute $w_1[l], w_2[l], w_3[l]$ for $l \leftarrow 0, \ldots, P-1$ and store in shared memory.
2: **end parfor**
3: ___TeamBarrier( )
4: **parfor** $t_x \leftarrow 0, \ldots, (P/2)^3 - 1$ **do** // TeamThreadRange over $(P/2)^3$ threads
5: **for** $\alpha \in$ `Colleagues`(β) **do**
6: $\boldsymbol{g}_\ell \leftarrow$ Index2Grid(t_x, α) // Assign $\boldsymbol{g}_\ell$ to a thread
7: **for** $\boldsymbol{y}_j \in \beta$ **do**
8: LoadScratch$(w_1[(\boldsymbol{g}_\ell - \boldsymbol{a}_j)_1], w_2[(\boldsymbol{g}_\ell - \boldsymbol{a}_j)_2], w_3[(\boldsymbol{g}_\ell - \boldsymbol{a}_j)_3], \boldsymbol{f}(\boldsymbol{y}_j))$
9: $\boldsymbol{\phi}_{\text{loc}} \leftarrow \boldsymbol{\phi}_{\text{loc}} + w_1\, w_2\, w_3\, \boldsymbol{f}(\boldsymbol{y}_j)$
10: **end for**
11: AtomicAdd$(\boldsymbol{\phi}(\boldsymbol{g}_\ell), \boldsymbol{\phi}_{\text{loc}})$ // $\mathcal{O}(27N_s)$ atomics
12: **end for**
13: **end parfor**

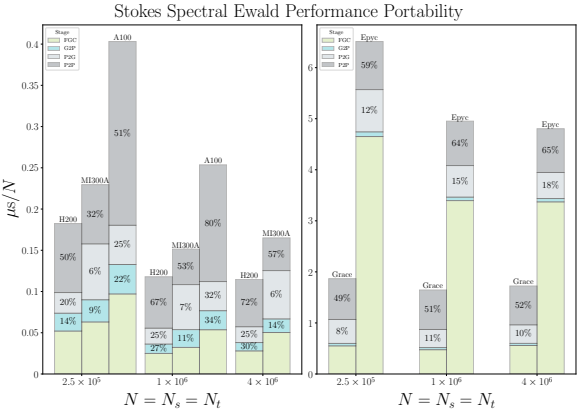


Figure: Spectral Ewald run on multiple machines (lower is better). $E = 10^{-9}$, $s = 224$.

Table: Runtime in *milliseconds* for multi-GPU weak scaling test with N_{GPU} NVIDIA H200 GPUs, MPI communication, and NVIDIA's cuFFTMp library. $E = 10^{-9}$, $s = 224$, $N = 4 \times 10^6 N_{\mathrm{GPU}}$.

Step	$N_{\mathrm{GPU}} = 1$	2	4	8	16
P2P	217.7	223.8	225.4	226.9	226.3
P2G	72.2	75.7	75.7	75.7	75.5
FFT	46.1	581.8	737.4	772.2	686.1
IFFT	1.5	24.0	28.6	30.7	26.1
G2P	39.9	40.6	40.6	40.4	40.7
MPI-SORT	5.6	22.7	23.2	36.4	61.7
MPI-GHOST-SOURCE	0.4	6.8	7.0	6.6	6.4
MPI-GHOST-GRID	0.2	3.2	5.2	4.7	4.6