Autotuning multigrid parameters in the HMC on different architectures

Marco Garofalo, Bartosz Kostrzewa, Simone Romiti, Aniket Sen

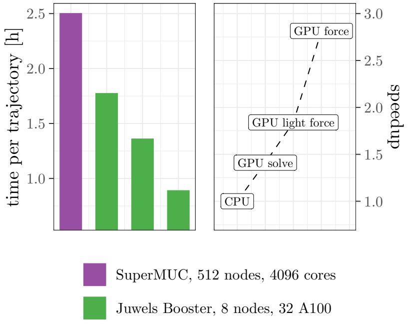
Rheinische Friedrich-Wilhelms-Universität Bonn

Lattice 2024, Liverpool, United Kingdom

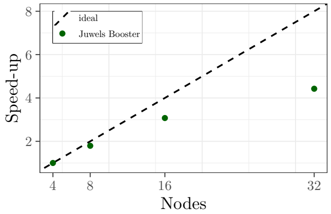
tmLQCD + QUDA

improvements over the last 18 months

$64^3 \cdot 128$ at M_π^{phys} (Juwels Booster)



Juwels Booster (A100) strong scaling

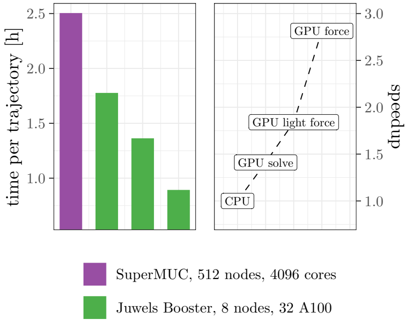


- tmLQCD: ETMC workhorse HMC implementation for $N_f = 2+1+1$ twisted mass Wilson (clover) simulations

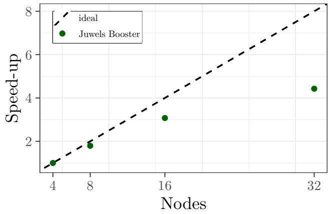
tmLQCD + QUDA

improvements over the last 18 months

$64^3 \cdot 128$ at M_π^{phys} (Juwels Booster)



Juwels Booster (A100) strong scaling

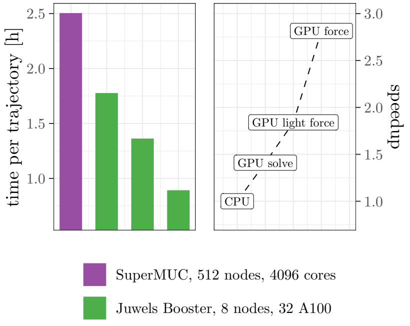


- tmLQCD: ETMC workhorse HMC implementation for $N_f = 2+1+1$ twisted mass Wilson (clover) simulations
- CPU $\rightarrow$ GPU speedup: up to 2.8 in real time as offloading fraction increases

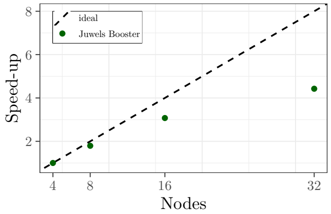
tmLQCD + QUDA

improvements over the last 18 months

$64^3 \cdot 128$ at M_π^{phys} (Juwels Booster)



Juwels Booster (A100) strong scaling

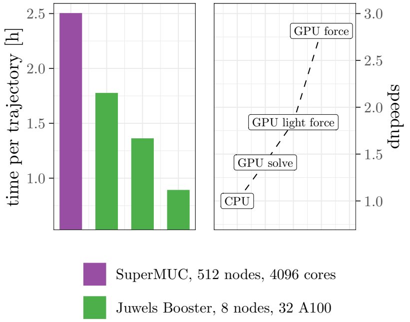


- tmLQCD: ETMC workhorse HMC implementation for $N_f = 2+1+1$ twisted mass Wilson (clover) simulations
- CPU $\to$ GPU speedup: up to 2.8 in real time as offloading fraction increases
- improvement in terms of energy cost much higher

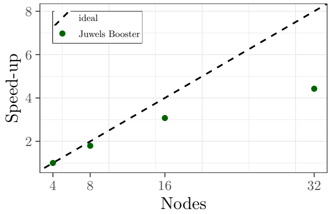
tmLQCD + QUDA

improvements over the last 18 months

$64^3 \cdot 128$ at M_π^{phys} (Juwels Booster)



Juwels Booster (A100) strong scaling

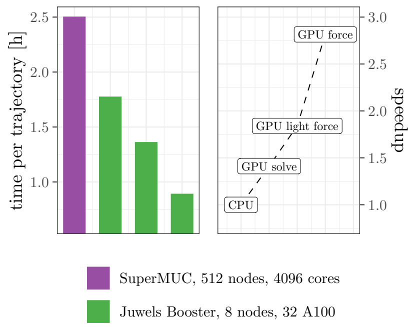


- tmLQCD: ETMC workhorse HMC implementation for $N_f = 2+1+1$ twisted mass Wilson (clover) simulations
- CPU $\rightarrow$ GPU speedup: up to 2.8 in real time as offloading fraction increases
- improvement in terms of energy cost much higher
- GPU utilisation $>$70% and even up to 90% when many CPU cores are available

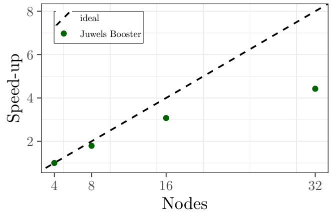
tmLQCD + QUDA

improvements over the last 18 months

$64^3 \cdot 128$ at M_π^{phys} (Juwels Booster)



Juwels Booster (A100) strong scaling

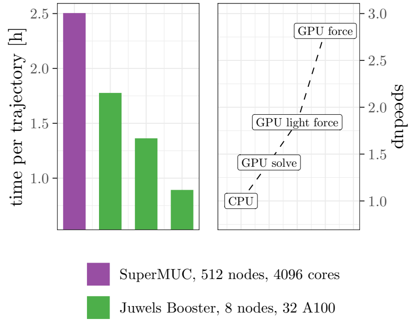


- tmLQCD: ETMC workhorse HMC implementation for $N_f = 2 + 1 + 1$ twisted mass Wilson (clover) simulations
- CPU $\to$ GPU speedup: up to 2.8 in real time as offloading fraction increases
- improvement in terms of energy cost much higher
- GPU utilisation $>$70% and even up to 90% when many CPU cores are available
- also offloaded: gradient flow, online eigenvalue and correlator measurements

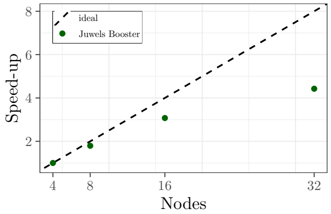
tmLQCD + QUDA

improvements over the last 18 months

$64^3 \cdot 128$ at M_π^{phys} (Juwels Booster)

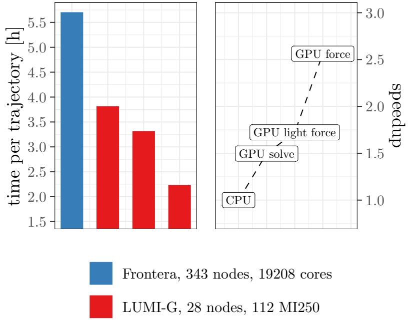


Juwels Booster (A100) strong scaling

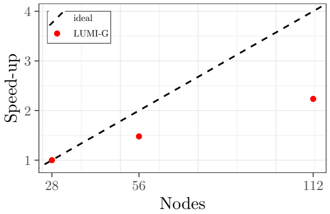


- tmLQCD: ETMC workhorse HMC implementation for $N_f = 2 + 1 + 1$ twisted mass Wilson (clover) simulations
- CPU $\rightarrow$ GPU speedup: up to 2.8 in real time as offloading fraction increases
- improvement in terms of energy cost much higher
- GPU utilisation >70% and even up to 90% when many CPU cores are available
- also offloaded: gradient flow, online eigenvalue and correlator measurements

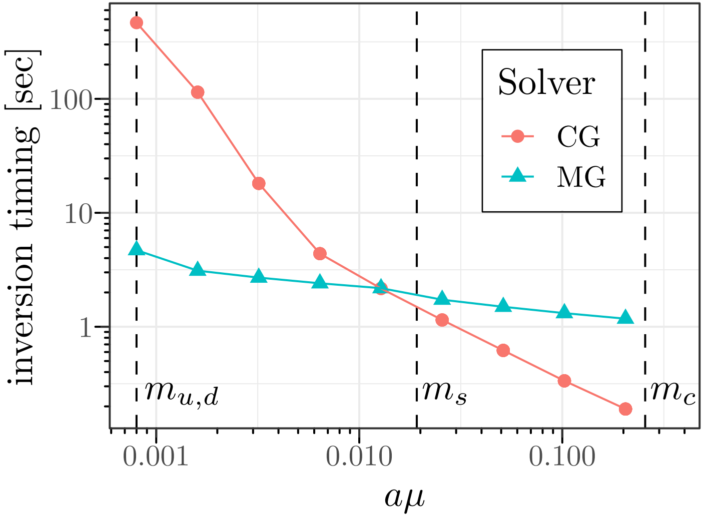
$112^3 \cdot 224$ at M_π^{phys} (LUMI-G)



LUMI-G (MI250) strong scaling



MG solver in the light sector



Comparison between MG-preconditioned-GCR
mixed-precision CG (GPU)
MG timing: two inversions + unavoidable overheads from coarse operator updates between D and $D^\dagger$ inversions

Light sector of MD Hamiltonian

In practice we employ

- 2 to 3 ρ-shifts (shifting the EO-operator)
- 3-4 time scales

$\rightarrow$ per trajectory need to solve systems with:

- $\rho = 0$ about $\mathcal{O}(100)$ times $\rightarrow$ MG
- $\rho \approx 0.001$ about $\mathcal{O}(100)$ times $\rightarrow$ MG
- $\rho \approx 0.01$ about $\mathcal{O}(200)$ times $\rightarrow$ CG
- $\rho \approx 0.1$ about $\mathcal{O}(400)$ times $\rightarrow$ CG

MG requires two solves in derivative and an update of the coarse operator (due to twisted mass sign change), but easily wins up to $\rho \approx am_s$.

Moving MG parameters from one machine to another

- Late 2022 / early 2023
  - started production of $112^3 \cdot 224$ physical point ensemble on LUMI-G (MI250)
- First computing time estimates based on performance on Juwels Booster (A100).
- Intermediate grid: $56 \cdot 4 \cdot 4 \cdot 8$ per GPU
- Coarsest grid: $8 \cdot 2 \cdot 2 \cdot 4$ per GPU
- MG parameters taken from experience

Moving MG parameters from one machine to another

- Late 2022 / early 2023
  - started production of $112^3 \cdot 224$ physical point ensemble on LUMI-G (MI250)
- First computing time estimates based on performance on Juwels Booster (A100).
- Intermediate grid: $56 \cdot 4 \cdot 4 \cdot 8$ per GPU
- Coarsest grid: $8 \cdot 2 \cdot 2 \cdot 4$ per GPU
- MG parameters taken from experience

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	1.0	1.0	50.0
`mg-coarse-solver-tol`	0.1	0.1	0.1
`mg-coarse-solver-maxiter`	100	100	100
`mg-nu-post`	4	4	4
`mg-nu-pre`	2	2	2
`mg-smoother-tol`	0.1	0.1	0.1
`mg-omega`	0.9	0.9	0.9

Moving MG parameters from one machine to another

- Late 2022 / early 2023
  - started production of $112^3 \cdot 224$ physical point ensemble on LUMI-G (MI250)
- First computing time estimates based on performance on Juwels Booster (A100).
- Intermediate grid: $56 \cdot 4 \cdot 4 \cdot 8$ per GPU
- Coarsest grid: $8 \cdot 2 \cdot 2 \cdot 4$ per GPU
- MG parameters taken from experience

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	1.0	1.0	50.0
`mg-coarse-solver-tol`	0.1	0.1	0.1
`mg-coarse-solver-maxiter`	100	100	100
`mg-nu-post`	4	4	4
`mg-nu-pre`	2	2	2
`mg-smoother-tol`	0.1	0.1	0.1
`mg-omega`	0.9	0.9	0.9

Juwels Booster

28 nodes: $\sim$ 6 seconds / solve

LUMI-G

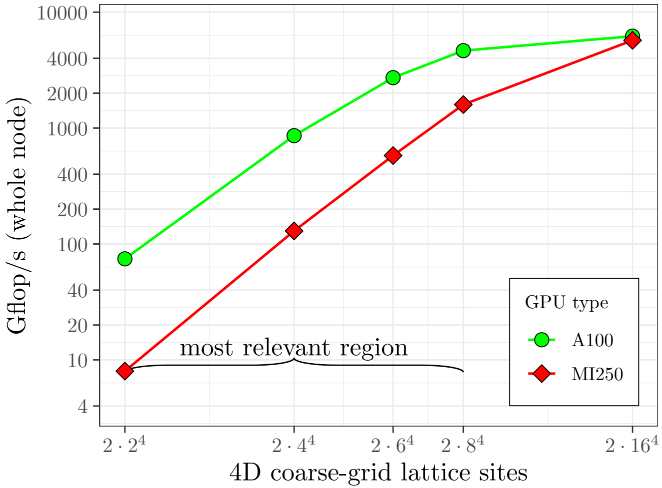
28 nodes

- **in 2023: $\sim$ 41 seconds / solve**
- today: $\sim$ 14 seconds / solve

Origin of the performance difference?

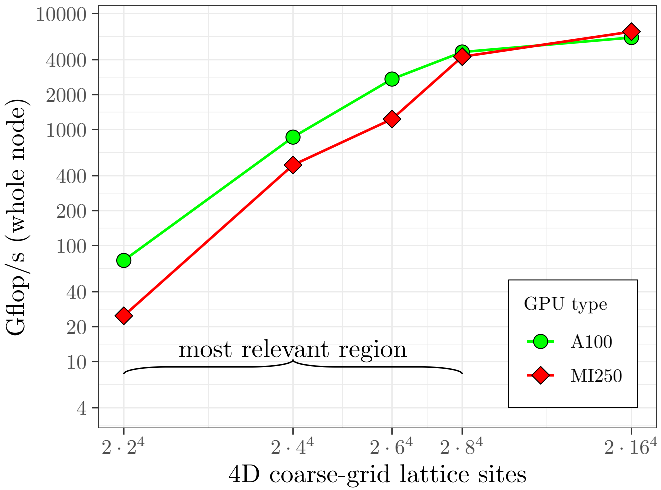
QUDA coarse-grid operator benchmark (single precision, 24 colours)

early 2023: full node (4 A100 GPUs / 8 MI250 GCDs)



- back in 2023: up to factor 6 difference in coarse-grid operator performance

today: full node (4 A100 GPUs / 8 MI250 GCDs)

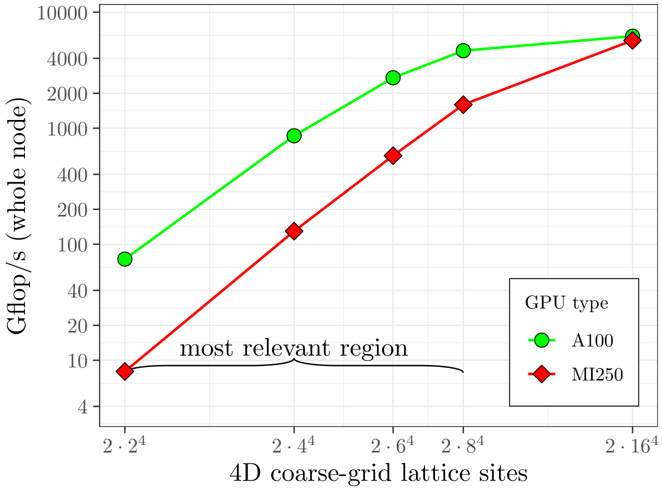


- today: down to about a factor of 2 difference

Origin of the performance difference?

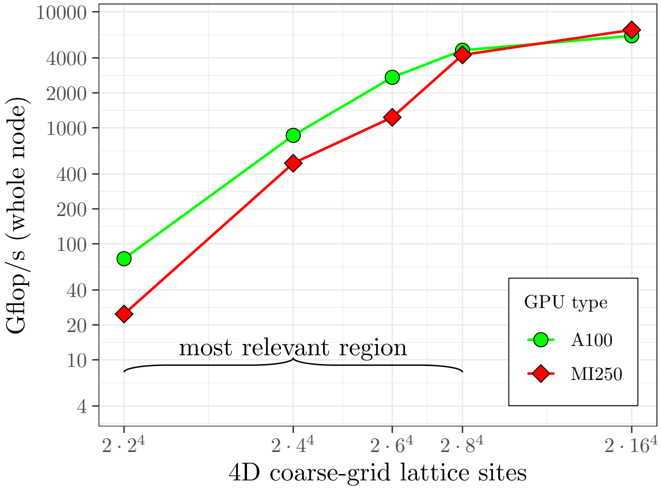
QUDA coarse-grid operator benchmark (single precision, 24 colours)

early 2023: full node (4 A100 GPUs / 8 MI250 GCDs)



- back in 2023: up to factor 6 difference in coarse-grid operator performance

today: full node (4 A100 GPUs / 8 MI250 GCDs)



- today: down to about a factor of 2 difference

Maybe we can find a different balance between coarse, intermediate and fine iterations to obtain better performance on MI250?

Exhaustive search of MG parameters?

Parameters that can be tuned w/out redoing MG setup

parameter	sensible choices
`mg-mu-factor`	$5 \cdot 5 \cdot 15 = 375$
`mg-coarse-solver-tol`	$4^2 = 16$
`mg-coarse-solver-maxiter`	$4^2 = 16$
`mg-nu-post`	$4^3 = 64$
`mg-nu-pre`	$4^3 = 64$
`mg-smoother-tol`	$3^3 = 9$
`mg-omega`	$3^3 = 9$
total	$\approx 10^{10}$ combs

Exhaustive search of MG parameters?

Parameters which require redoing MG setup

parameter	sensible choices
`mg-block-size`	$\approx 3^2 = 9$
`mg-nvec`	$\approx 2^2 = 4$
total	≈ 36 combs

Parameters that can be tuned w/out redoing MG setup

parameter	sensible choices
`mg-mu-factor`	$5 \cdot 5 \cdot 15 = 375$
`mg-coarse-solver-tol`	$4^2 = 16$
`mg-coarse-solver-maxiter`	$4^2 = 16$
`mg-nu-post`	$4^3 = 64$
`mg-nu-pre`	$4^3 = 64$
`mg-smoother-tol`	$3^3 = 9$
`mg-omega`	$3^3 = 9$
total	$\approx 10^{10}$ combs

Exhaustive search of MG parameters?

Parameters that can be tuned w/out redoing MG setup

parameter	sensible choices
`mg-mu-factor`	$5 \cdot 5 \cdot 15 = 375$
`mg-coarse-solver-tol`	$4^2 = 16$
`mg-coarse-solver-maxiter`	$4^2 = 16$
`mg-nu-post`	$4^3 = 64$
`mg-nu-pre`	$4^3 = 64$
`mg-smoother-tol`	$3^3 = 9$
`mg-omega`	$3^3 = 9$
total	$\approx 10^{10}$ combs

Parameters which require redoing MG setup

parameter	sensible choices
`mg-block-size`	$\approx 3^2 = 9$
`mg-nvec`	$\approx 2^2 = 4$
total	≈ 36 combs

Fully exhaustive search clearly not possible!

- fix certain params on certain levels
- do not tune less relevant params
- tune mostly / only on coarser levels

Can restrict to relevant subset of $\approx 10^6$ parameter combinations or so.

Still a major investment of computing time!

Exhaustive search of MG parameters?

Parameters that can be tuned w/out redoing MG setup

parameter	sensible choices
`mg-mu-factor`	$5 \cdot 5 \cdot 15 = 375$
`mg-coarse-solver-tol`	$4^2 = 16$
`mg-coarse-solver-maxiter`	$4^2 = 16$
`mg-nu-post`	$4^3 = 64$
`mg-nu-pre`	$4^3 = 64$
`mg-smoother-tol`	$3^3 = 9$
`mg-omega`	$3^3 = 9$
total	$\approx 10^{10}$ combs

Parameters which require redoing MG setup

parameter	sensible choices
`mg-block-size`	$\approx 3^2 = 9$
`mg-nvec`	$\approx 2^2 = 4$
total	≈ 36 combs

Fully exhaustive search clearly not possible!

- fix certain params on certain levels
- do not tune less relevant params
- tune mostly / only on coarser levels

Can restrict to relevant subset of $\approx 10^6$ parameter combinations or so.

Still a major investment of computing time!

Can we we use our intuition to find good MG setups more quickly?

“Automatic” tuning of MG parameters

https://github.com/etmc/tmLQCD/tree/deriv_mg_tune

Ideas behind proceduce:

- Always start at coarsest grid.
- Tune most relevant params first.
- Ignore small fluctuations.
- Accept even small improvements (might need several steps to see benefit).
- Tune on multiple gauge configurations.

“Automatic” tuning of MG parameters

https://github.com/etmc/tmLQCD/tree/deriv_mg_tune

Ideas behind proceduce:

- Always start at coarsest grid.
- Tune most relevant params first.
- Ignore small fluctuations.
- Accept even small improvements (might need several steps to see benefit).
- Tune on multiple gauge configurations.

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	μ_0	μ_1	μ_2
`mg-coarse-solver-tol`	r_0	r_1	r_2
`mg-coarse-solver-maxiter`	n_0	n_1	n_2
`mg-nu-post`	ν_0^{post}	ν_1^{post}	ν_2^{post}
`mg-nu-pre`	ν_0^{pre}	ν_1^{pre}	ν_2^{pre}
`mg-smoother-tol`	r_0^s	r_1^s	r_2^s
`mg-omega`	ω_0	ω_1	ω_2

“Automatic” tuning of MG parameters

https://github.com/etmc/tmLQCD/tree/deriv_mg_tune

Ideas behind proceduce:

- Always start at coarsest grid.
- Tune most relevant params first.
- Ignore small fluctuations.
- Accept even small improvements (might need several steps to see benefit).
- Tune on multiple gauge configurations.
- **How to deal with non-converging solves?**

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	μ_0	μ_1	μ_2
`mg-coarse-solver-tol`	r_0	r_1	r_2
`mg-coarse-solver-maxiter`	n_0	n_1	n_2
`mg-nu-post`	ν_0^{post}	ν_1^{post}	ν_2^{post}
`mg-nu-pre`	ν_0^{pre}	ν_1^{pre}	ν_2^{pre}
`mg-smoother-tol`	r_0^s	r_1^s	r_2^s
`mg-omega`	ω_0	ω_1	ω_2

“Automatic” tuning of MG parameters

Details

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	μ_0	μ_1	μ_2
`mg-coarse-solver-tol`	r_0	r_1	r_2
`mg-coarse-solver-maxiter`	n_0	n_1	n_2
`mg-nu-post`	ν_0^{post}	ν_1^{post}	ν_2^{post}
`mg-nu-pre`	ν_0^{pre}	ν_1^{pre}	ν_2^{pre}
`mg-smoother-tol`	r_0^s	r_1^s	r_2^s
`mg-omega`	ω_0	ω_1	ω_2

repeat μ_2

Global tuning procedure parameters

- Number of tuning steps per gauge configuration, N (f.ex. 100).
- Tolerance δ: stop tuning the current parameter, f.ex. $t_i/t_{\text{best}} > 0.995$
- Threshold ρ: ignore fluctuations when choosing t_{best}, f.ex. $t_i/t_{\text{best}} > 0.999$

“Automatic” tuning of MG parameters

Details

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	μ_0	μ_1	μ_2
`mg-coarse-solver-tol`	r_0	r_1	r_2
`mg-coarse-solver-maxiter`	n_0	n_1	n_2
`mg-nu-post`	ν_0^{post}	ν_1^{post}	ν_2^{post}
`mg-nu-pre`	ν_0^{pre}	ν_1^{pre}	ν_2^{pre}
`mg-smoother-tol`	r_0^s	r_1^s	r_2^s
`mg-omega`	ω_0	ω_1	ω_2

repeat μ_2

Global tuning procedure parameters

- Number of tuning steps per gauge configuration, N (f.ex. 100).
- Tolerance δ: stop tuning the current parameter, f.ex. $t_i/t_{\text{best}} > 0.995$
- Threshold ρ: ignore fluctuations when choosing t_{best}, f.ex. $t_i/t_{\text{best}} > 0.999$

For each parameter p on level ℓ

- maximum number of steps to be done n_p^ℓ
- change in parameter per step $\pm\Delta_p^\ell$

1 perform n_p^ℓ steps of $p^\ell + \Delta_p^\ell$, or until timing stops improving

2 if timing does not improve (or worsens), move to next p on current ℓ

3 move to next-finest level and follow same sequence

4 if step $i < N$, go back to (1)

4 if step $i = N$, move to next gauge configuration, reset $i = 0$

Tuning MG parameters for a $112^3 \cdot 224$ ensemble at M_π^{phys} (from above, LUMI-G)

tuning from above
reducing cost step-by-step

Initial parameters:

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	1.0	1.0	30.0
`mg-coarse-solver-tol`	0.05	0.05	0.05
`mg-coarse-solver-maxiter`	50	50	50
`mg-nu-post`	6	6	6
`mg-nu-pre`	6	6	6
`mg-smoother-tol`	0.05	0.05	0.05
`mg-omega`	0.85	0.85	0.85

- Positive Δ for:
  - `mg-mu-factor`
  - `mg-coarse-solver-tol`
  - `mg-smoother-tol`
- Negative Δ for:
  - `mg-coarse-solver-maxiter`
  - `mg-nu-post`
  - `mg-nu-pre`

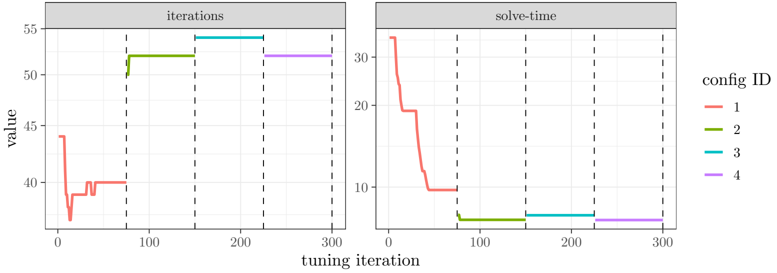
Tuning MG parameters for a $112^3 \cdot 224$ ensemble at M_π^{phys} (from above, LUMI-G)

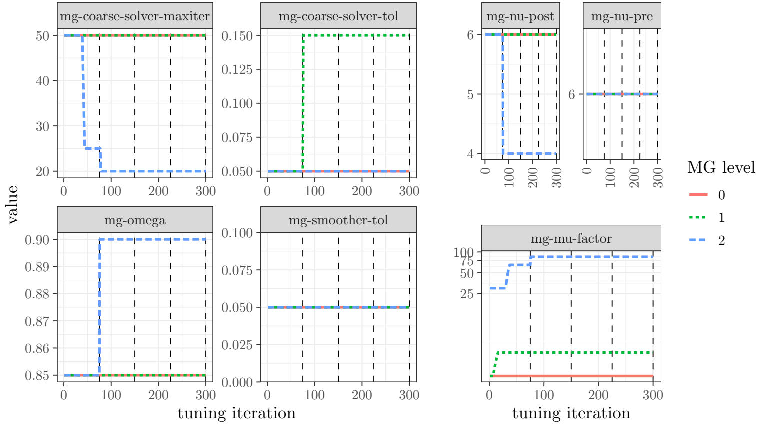
tuning from above
reducing cost step-by-step

Initial parameters:

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	1.0	1.0	30.0
`mg-coarse-solver-tol`	0.05	0.05	0.05
`mg-coarse-solver-maxiter`	50	50	50
`mg-nu-post`	6	6	6
`mg-nu-pre`	6	6	6
`mg-smoother-tol`	0.05	0.05	0.05
`mg-omega`	0.85	0.85	0.85

- Positive Δ for:
  - `mg-mu-factor`
  - `mg-coarse-solver-tol`
  - `mg-smoother-tol`
- Negative Δ for:
  - `mg-coarse-solver-maxiter`
  - `mg-nu-post`
  - `mg-nu-pre`





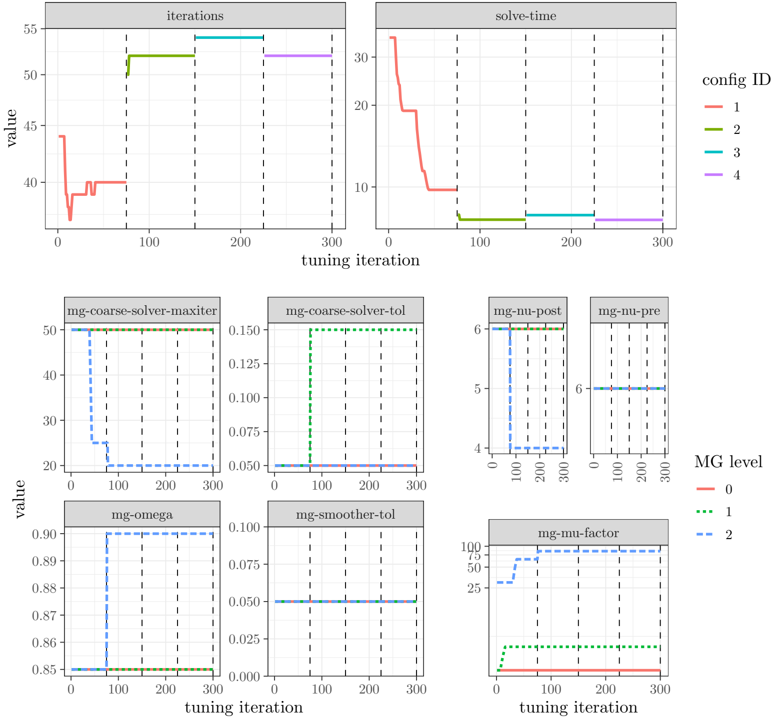
Tuning MG parameters for a $112^3 \cdot 224$ ensemble at M_π^{phys} (from above, LUMI-G)

tuning from above
reducing cost step-by-step

Initial parameters:

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	1.0	1.0	30.0
`mg-coarse-solver-tol`	0.05	0.05	0.05
`mg-coarse-solver-maxiter`	50	50	50
`mg-nu-post`	6	6	6
`mg-nu-pre`	6	6	6
`mg-smoother-tol`	0.05	0.05	0.05
`mg-omega`	0.85	0.85	0.85

- Positive Δ for:
  - `mg-mu-factor`
  - `mg-coarse-solver-tol`
  - `mg-smoother-tol`
- Negative Δ for:
  - `mg-coarse-solver-maxiter`
  - `mg-nu-post`
  - `mg-nu-pre`



from 40+ seconds to $\sim$ 8 seconds
in a few hundred solves

Tuning MG parameters for a $112^3 \cdot 224$ ensemble at M_π^{phys} (from below, LUMI-G)

tuning from below
increasing cost step-by-step

Initial parameters:

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	1.0	1.0	30.0
`mg-coarse-solver-tol`	0.55	0.55	0.55
`mg-coarse-solver-maxiter`	5	5	5
`mg-nu-post`	1	1	1
`mg-nu-pre`	1	1	1
`mg-smoother-tol`	0.55	0.55	0.55
`mg-omega`	0.85	0.85	0.85

- Positive Δ for:
  - `mg-mu-factor`
  - `mg-coarse-solver-maxiter`
  - `mg-nu-post`
  - `mg-nu-pre`
- Negative Δ for:
  - `mg-coarse-solver-tol`
  - `mg-smoother-tol`

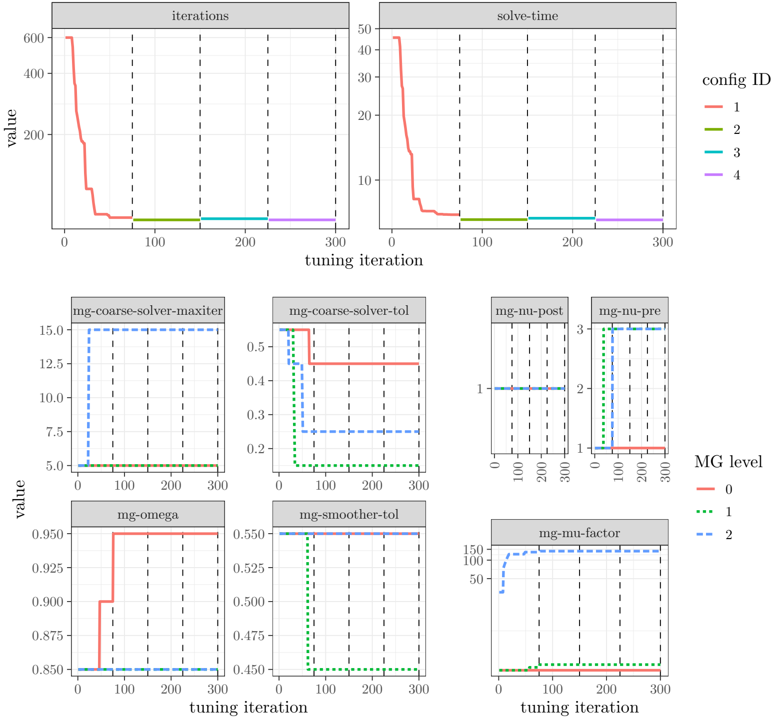
Tuning MG parameters for a $112^3 \cdot 224$ ensemble at M_π^{phys} (from below, LUMI-G)

tuning from below
increasing cost step-by-step

Initial parameters:

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	1.0	1.0	30.0
`mg-coarse-solver-tol`	0.55	0.55	0.55
`mg-coarse-solver-maxiter`	5	5	5
`mg-nu-post`	1	1	1
`mg-nu-pre`	1	1	1
`mg-smoother-tol`	0.55	0.55	0.55
`mg-omega`	0.85	0.85	0.85

- Positive Δ for:
  - `mg-mu-factor`
  - `mg-coarse-solver-maxiter`
  - `mg-nu-post`
  - `mg-nu-pre`
- Negative Δ for:
  - `mg-coarse-solver-tol`
  - `mg-smoother-tol`



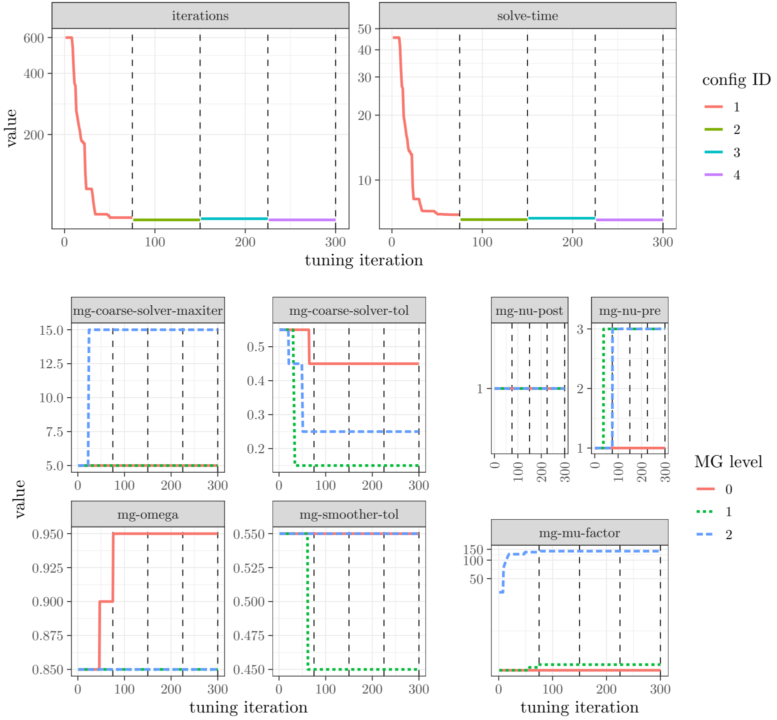
Tuning MG parameters for a $112^3 \cdot 224$ ensemble at M_π^{phys} (from below, LUMI-G)

tuning from below
increasing cost step-by-step

Initial parameters:

parameter	lvl 0	lvl 1	lvl 2
`mg-mu-factor`	1.0	1.0	30.0
`mg-coarse-solver-tol`	0.55	0.55	0.55
`mg-coarse-solver-maxiter`	5	5	5
`mg-nu-post`	1	1	1
`mg-nu-pre`	1	1	1
`mg-smoother-tol`	0.55	0.55	0.55
`mg-omega`	0.85	0.85	0.85

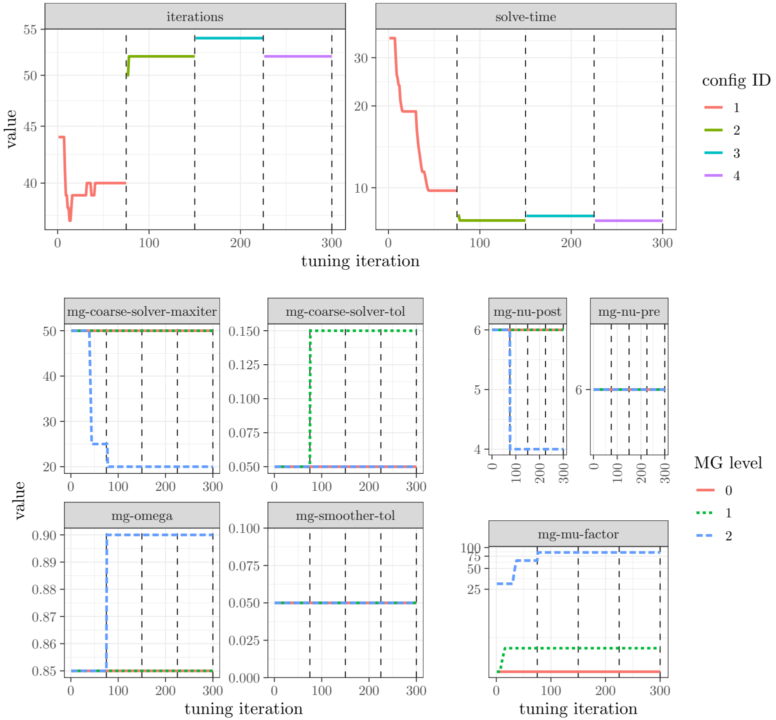
- Positive Δ for:
  - `mg-mu-factor`
  - `mg-coarse-solver-maxiter`
  - `mg-nu-post`
  - `mg-nu-pre`
- Negative Δ for:
  - `mg-coarse-solver-tol`
  - `mg-smoother-tol`



from non-convergence to $\sim$ 7 seconds
in a few hundred solves

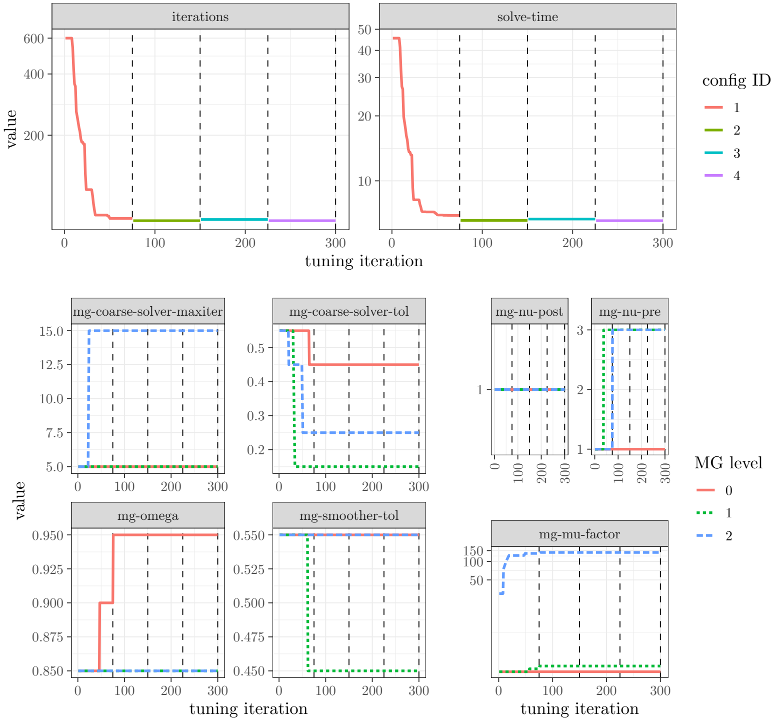
Tuning MG parameters for a $112^3 \cdot 224$ ensemble at M_π^{phys} (Comparison)

tuning from above



7.5 to 9 seconds

tuning from below



6.5 to 7 seconds

(depending on gauge configuration)

So, does it work?

Let's recall the $112^3 \cdot 224$ ensemble @ M_π^{phys} running on LUMI-G.

Juwels Booster

28 nodes

- Before tuning: $\sim$ 6 seconds / solve
- After tuning: $\sim$ 4 seconds / solve

LUMI-G

28 nodes

- Before tuning: $\sim$ 14 seconds / solve
- After tuning: $\sim$ 7 seconds / solve

So, does it work?

Let's recall the $112^3 \cdot 224$ ensemble @ M_π^{phys} running on LUMI-G.

Juwels Booster

28 nodes

- Before tuning: $\sim$ 6 seconds / solve
- After tuning: $\sim$ 4 seconds / solve

LUMI-G

28 nodes

- Before tuning: $\sim$ 14 seconds / solve
- After tuning: $\sim$ 7 seconds / solve

This was way more impressive back in 2023 when we went from 41 to $\sim$ 10 seconds.

So, does it work?

Let's recall the $112^3 \cdot 224$ ensemble @ M_π^{phys} running on LUMI-G.

Juwels Booster

28 nodes

- Before tuning: $\sim$ 6 seconds / solve
- After tuning: $\sim$ 4 seconds / solve

LUMI-G

28 nodes

- Before tuning: $\sim$ 14 seconds / solve
- After tuning: $\sim$ 7 seconds / solve

This was way more impressive back in 2023 when we went from 41 to $\sim$ 10 seconds.

Quickly finds acceptable MG setups for the HMC.

So, does it work?

Let's recall the $112^3 \cdot 224$ ensemble @ M_π^{phys} running on LUMI-G.

Juwels Booster

28 nodes

- Before tuning: $\sim$ 6 seconds / solve
- After tuning: $\sim$ 4 seconds / solve

LUMI-G

28 nodes

- Before tuning: $\sim$ 14 seconds / solve
- After tuning: $\sim$ 7 seconds / solve

This was way more impressive back in 2023 when we went from 41 to $\sim$ 10 seconds.

Quickly finds acceptable MG setups for the HMC.

Improves setups also in situations where intuition was okay.

So, does it work?

Let's recall the $112^3 \cdot 224$ ensemble @ M_π^{phys} running on LUMI-G.

Juwels Booster

28 nodes

- Before tuning: $\sim$ 6 seconds / solve
- After tuning: $\sim$ 4 seconds / solve

LUMI-G

28 nodes

- Before tuning: $\sim$ 14 seconds / solve
- After tuning: $\sim$ 7 seconds / solve

This was way more impressive back in 2023 when we went from 41 to $\sim$ 10 seconds.

Quickly finds acceptable MG setups for the HMC.

Improves setups also in situations where intuition was okay.

Finds parameters combinations that we would not have thought of.

So, does it work?

Let's recall the $112^3 \cdot 224$ ensemble @ M_π^{phys} running on LUMI-G.

Juwels Booster

28 nodes

- Before tuning: $\sim$ 6 seconds / solve
- After tuning: $\sim$ 4 seconds / solve

LUMI-G

28 nodes

- Before tuning: $\sim$ 14 seconds / solve
- After tuning: $\sim$ 7 seconds / solve

This was way more impressive back in 2023 when we went from 41 to $\sim$ 10 seconds.

Quickly finds acceptable MG setups for the HMC.

Improves setups also in situations where intuition was okay.

Finds parameters combinations that we would not have thought of.

Nice corollary: can also further improve coarse-grid-deflated solver, used to good effect on LUMI-G.

$\Rightarrow$ Useful also for measurement campaigns.

Limitations

- Not currently integrated into HMC.
  - Currently: need to prepare set of configurations and perform separate run.
  - Integration directly into HMC possible: tuner is already called from within fermionic derivative.
- Not tested on untwisted Wilson clover.
  - Should work out of the box, need gauge configs in ILDG format. (ignore `mg-mu-factor`)
- Does not tune parameters which need MG setup to be regenerated.
  - Required logic extension simple but tedious.
- Some of the parameter evolution does not seem to make a lot of sense.
  - Might require some more fine-tuned intervention logic.
- Thresholds and starting parameters can have big impact on tuning quality.
  - Need some more systematic guidelines to judge impact of lattice spacing, target quark mass and lattice volume.

Where to get it?

If you want to play around with the code:

- `https://github.com/etmc/tmLQCD/tree/deriv_mg_tune`
- `deriv_mg_tune` executable
- all input file parameters explained in documentation
- `quda_interface.c`: `quda_mg_tune_params` function (and various helper functions)

Many thanks for your attention!

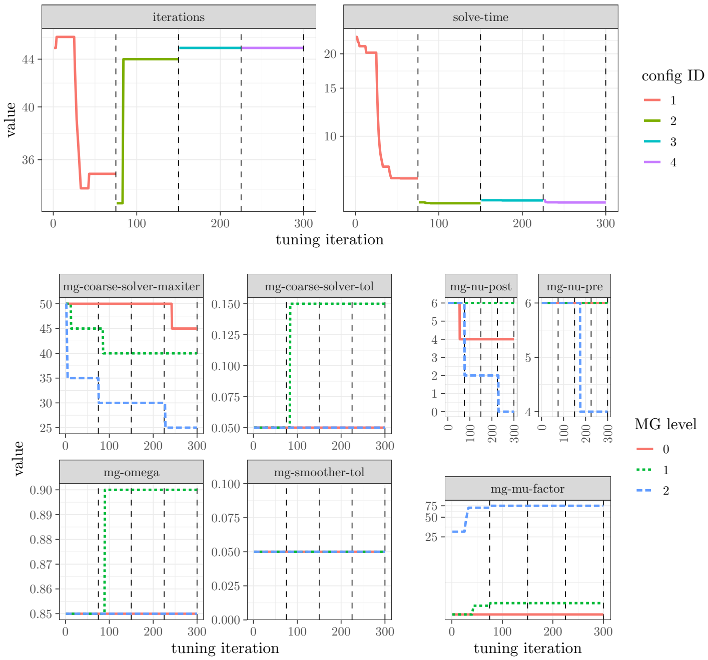
Backup

Backup Slides

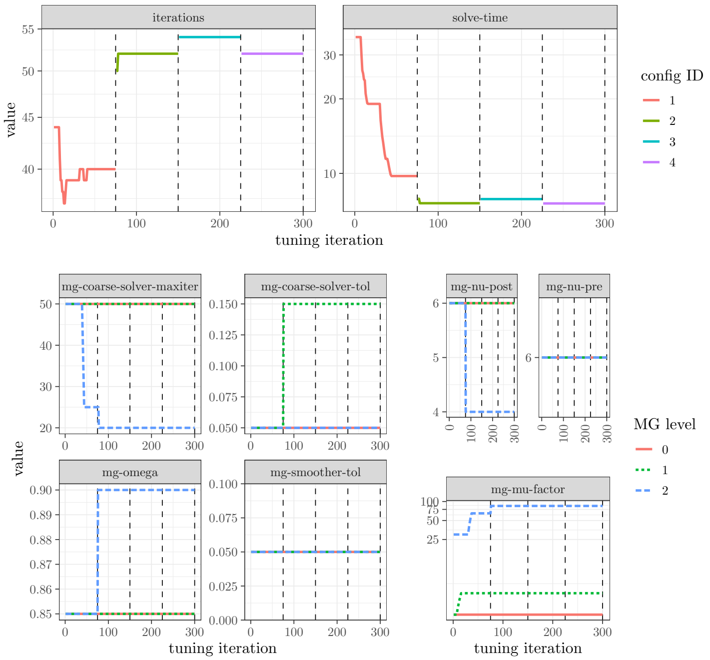
Comparison between Juwels Booster and LUMI-G

tuning from above

Juwels Booster



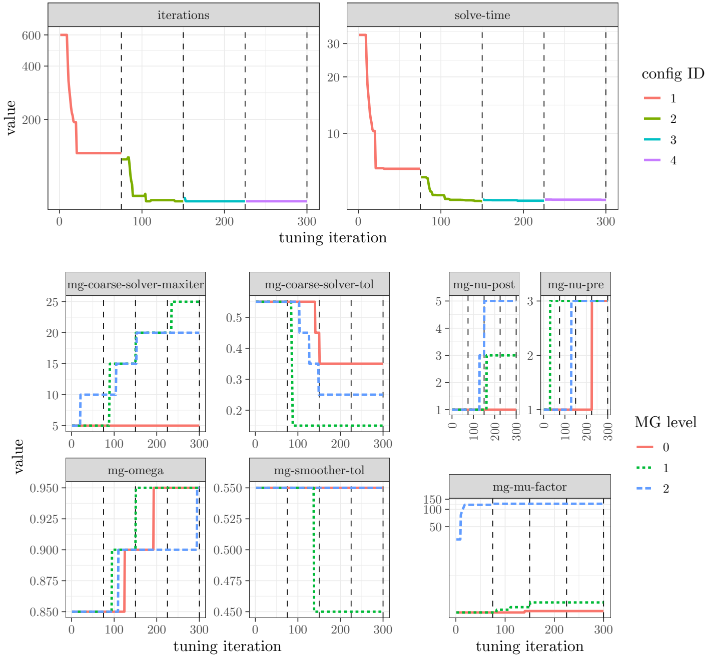
LUMI-G



Comparison between Juwels Booster and LUMI-G

tuning from below

Juwels Booster



LUMI-G