Optimizing Staggered All-to-all for GPUs

Michael Lynch

University of Illinois, Urbana-Champaign

Lattice 2023, FNAL

August 2, 2023

Why All-to-all?

- Approximates the exact propagator.

- Can calculate >2pt correlation functions without sequential solves.

- Useful in low-mode noise reduction techniques such as low mode averaging (LMA).

Constructing the Propagator

- Eigenvector Basis
  - *low modes* carry long-distance data[1]
  - $2N_e$ eigenvector pairs

$$M^{-1} \approx \sum_i^{2N_e} \frac{1}{\lambda_i} v_i v_i^\dagger$$

$$\not{D} = \begin{pmatrix} 0 & D_{eo} \\ D_{oe} & 0 \end{pmatrix}, \quad v_n^{(e)} = \frac{D_{eo} v_n^{(o)}}{i\tilde{\lambda}_n}, \quad v_n^{(o)} = \frac{D_{oe} v_n^{(e)}}{-i\tilde{\lambda}_n}.$$

$$M \begin{pmatrix} v_n^{(e)} \\ \pm v_n^{(o)} \end{pmatrix} = m \pm i\tilde{\lambda}_n \begin{pmatrix} v_n^{(e)} \\ \pm v_n^{(o)} \end{pmatrix}$$

[1]T. DeGrand, S. Schaefer. Comput. Phys. Commun. 159, 3 (2004).

Constructing the Propagator

- Stochastic Basis
  - N_r random sources

$$M^{-1} \approx \frac{1}{N_r} \sum_{i}^{N_r} \psi_i \eta_i^\dagger, \quad M\psi_i = \eta_i$$

$$\lim_{N_r \to \infty} \frac{1}{N_r} \sum_{i}^{N_r} \eta_i(y) \eta_i^\dagger(x) = \delta_{x,y} \mathbb{I}$$

- Hybrid Basis

$$M^{-1} \approx \sum_{i}^{N_e + N_r} u_i w_i^\dagger$$

$$\begin{aligned} w_i &\in \{v_1/\lambda_1, \ldots, v_{N_e}/\lambda_{N_e}, \psi_1 \ldots, \psi_{N_r}\}, \\ u_i &\in \{v_1, \ldots, v_{N_e}, \mathbb{P}\eta_1, \ldots, \mathbb{P}\eta_{N_r}\} \end{aligned}$$

$$\mathbb{P}\eta_i = \left(\mathbb{I} - \sum_{k=1}^{N_e} v_k v_k^\dagger \right) \eta_i .$$

J. Foley et al. Comput. Phys. Commun. 172, 145 (2005).

Constructing Meson Fields

- Example: connected two-point function

$$
\begin{aligned}
C^{\Gamma}(t) = & \sum_{\vec{x},\vec{y},t_0} tr\left\{M^{-1}(\vec{x},t_0;\vec{y},t+t_0)\Gamma M^{-1}(\vec{y},t+t_0;\vec{x},t_0)\Gamma\right\} \\
\approx & \sum_{t_0}\sum_{i,j}^{N_r+N_e}\sum_{\vec{x},\vec{y}} tr\left\{u_i(\vec{x},t_0)w_i^{\dagger}(\vec{y},t+t_0)\Gamma u_j(\vec{y},t+t_0)w_j^{\dagger}(\vec{x},t_0)\Gamma\right\} \\
\equiv & \sum_{t_0}\sum_{i,j}^{N}\mathcal{M}_{i,j}^{\Gamma}(t+t_0)\mathcal{M}_{i,j}^{\Gamma}(t_0).
\end{aligned}
$$

$$
\mathcal{M}_{i,j}^{\Gamma}(t) \equiv \sum_{\vec{x}} w_i^{\dagger}(\vec{x},t)\Gamma u_j(\vec{x},t)
$$

Starting Point - Grid+Hadrons Libraries

Original workflow:

1. Grid's Lanczos solver generates $\{v_i^{(o)}\}$

2. Hadrons builds full set, $\{w_i, v_j\}$

3. Grid kernel computes meson field in ${N_b}^2$ blocks on the CPU.
   - OMP threads work on separate time slices.
   - No communication needed between threads.

Kernel Changes - GPU offloading

- GPUs need more parallel tasks than CPUs.
- Still want to avoid dependencies between threads.
- Staggered fields are small enough to fit 100's of vectors on modern GPU memory.

➡ Parallelize over (w_i,v_j) :

```
accelerator_for2d(l_index,sizeL, r_index, sizeR, Nsimd, …
```

Kernel Changes - Eigenmode Kernel

- Create new kernel that accepts $\{v_i^{(o)}, v_j^{(e)}\}$ as parameters.
- Calculate two half-volume contractions instead of four full-volume.

$$
\begin{aligned}
\mathcal{M}^{\Gamma}_{2m,2n}(t) &= \mathcal{M}^{(e,e)\Gamma}_{m,n}(t) + \mathcal{M}^{(o,o)\Gamma}_{m,n}(t), \\
\mathcal{M}^{\Gamma}_{2m+1,2n}(t) &= \mathcal{M}^{(e,e)\Gamma}_{m,n}(t) - \mathcal{M}^{(o,o)\Gamma}_{m,n}(t), \\
\mathcal{M}^{\Gamma}_{2m,2n+1}(t) &= \mathcal{M}^{(e,e)\Gamma}_{m,n}(t) - \mathcal{M}^{(o,o)\Gamma}_{m,n}(t), \\
\mathcal{M}^{\Gamma}_{2m+1,2n+1}(t) &= \mathcal{M}^{(e,e)\Gamma}_{m,n}(t) + \mathcal{M}^{(o,o)\Gamma}_{m,n}(t).
\end{aligned}
$$

$$
\begin{aligned}
\lambda_{2n} &\equiv m + i\tilde{\lambda}_n, \\
\lambda_{2n+1} &\equiv m - i\tilde{\lambda}_n.
\end{aligned}
$$

Staggered Local Operators

- Fermion fields have no spin component.
- Spin is encoded in a phase parameter.

$$\mathcal{M}^{\Gamma}_{i,j}(t) = \sum_{\vec{x}} \phi_{\Gamma}(\vec{x},t) \left[w_i^{\dagger}(\vec{x},t) u_j(\vec{x},t) \right]$$

- Same inner product can be reused for multiple local operators.

Performance

V	N_e	N_r
$48^3 \times 64$	2000	0

System	Nodes	V_{local}	N_b	Gflops/s/GPU	t_{MF} (s)	$t_{CG_{\text{defl}}}$ (s)
Summit (V100)	8	$24^2 \times 16^2$	250	630	469	~ 25
Perlmutter (A100)	4	$24^3 \times 32^2$	500	1.1×10^3	250	~ 20

- Summit: Smaller N_b underutilizes GPUs but avoids thrashing.
- Low compute throughput for both
  - Consider using shared memory?

One-link Operator

$$J_\mu(r,r') = \alpha_\mu(r) U_\mu(r) \delta_{r+\hat{\mu},r'} + \text{H.C.},$$
$$\alpha_\mu(r) = (-1)^{r_\mu^<}$$

- Created class A2AWork
  - Maintains communication infrastructure between kernel calls
- Created new kernel
  - Takes link field parameter
- Created StagGamma class for general spin-taste ops.

Performance

V	N_e	N_r
$48^3 \times 64$	2000	0

System	Nodes	V_{local}	N_b	Gflops/s/GPU	t_{MF} (s)
Summit (V100)	8	$24^2 \times 16^2$	500	1.7×10^3	443
Perlmutter (A100)	4	$24^3 \times 32^2$	500	2.6×10^3	485

- Summit: Cache hit rates improve to >80%.
  - Actually faster than local operators
- Compute throughput still ~40%

Low-mode Averaging for Vector 2-point Functions

- Stochastic wall sources give good noise behavior at small t.
- Low modes give good noise behavior at large t.
- We want the best of both.

Component 1: Wall-to-all

$$C_{\mathrm{RW}}^{\Gamma}(t) \equiv \frac{1}{N_r}\sum_{i}^{N_r}\sum_{t_0,\vec{x}} tr\left[S_{\mathrm{RW},i}(\vec{x},t+t_0)\Gamma\gamma^5 S_{\mathrm{RW},i}^{\dagger}(\vec{x},t_0)\gamma^5\Gamma\right],$$

$$S_{\mathrm{RW},i}(\vec{x},t) = \sum_{y} M^{-1}(\vec{x},t;y)\eta_i(y).$$

Component 2: Low-mode Projection

$$C^{\Gamma}_{\mathrm{RW,LL}}(t) \equiv \frac{1}{N_r}\sum_{i}^{N_r}\sum_{t_0,\vec{x}} tr\left[S_{\mathrm{RWLL},i}(\vec{x},t+t_0)\Gamma\gamma^5 S^{\dagger}_{\mathrm{RWLL},i}(\vec{x},t_0)\gamma^5\Gamma\right],$$

$$S_{\mathrm{RWLL},i}(\vec{x},t) = \sum_{y}\sum_{j}^{N_e}\frac{1}{\lambda_j}v_j(\vec{x},t)v_j^{\dagger}(y)\eta_i(y).$$

Component 3: Low-mode All-to-all

$$C^{\Gamma}_{\mathrm{A2A,LL}}(t) = \frac{1}{N_t}\sum_{t_0}^{N_t}\sum_{i,j}^{N_e}\mathcal{M}^{\Gamma}_{i,j}(t)\mathcal{M}^{\Gamma}_{j,i}(t+t_0),$$

$$\mathcal{M}^{\Gamma}_{i,j}(t) \equiv \sum_{\vec{x}}\frac{1}{\lambda_i}v_i^{\dagger}(\vec{x},t)\Gamma v_j(\vec{x},t),$$

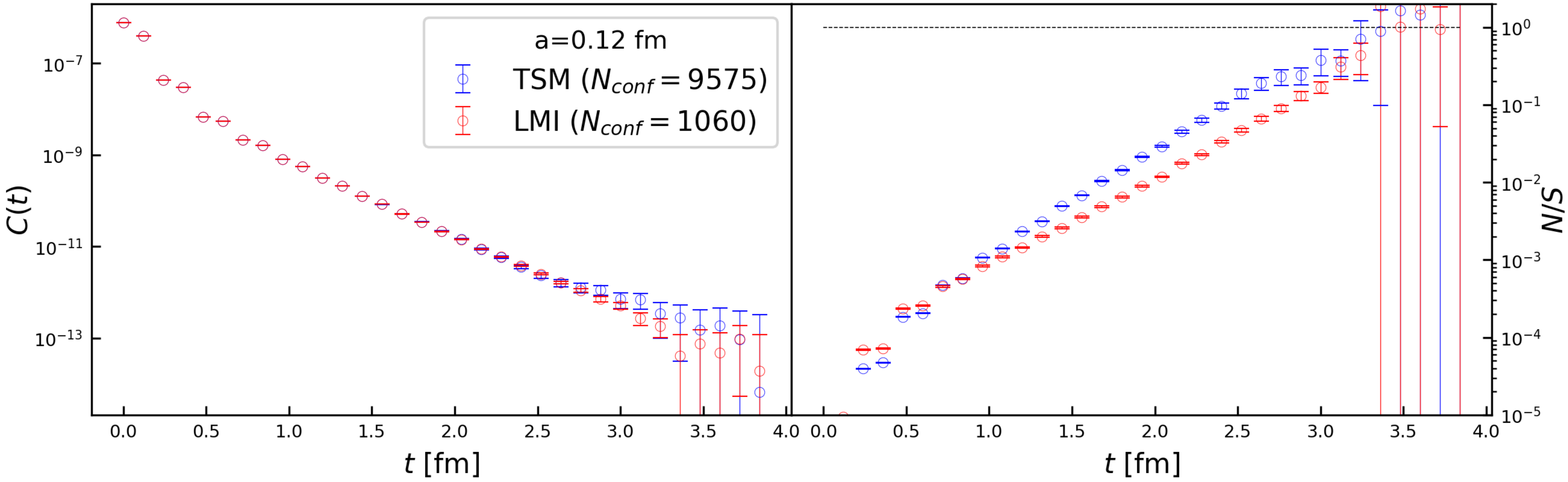
Result: Low-mode—improved Random Wall

$$C^{\Gamma}_{\mathrm{LMI}}(t) = C^{\Gamma}_{\mathrm{RW}}(t) - C^{\Gamma}_{\mathrm{RW,LL}}(t) + C^{\Gamma}_{\mathrm{A2A,LL}}(t)$$

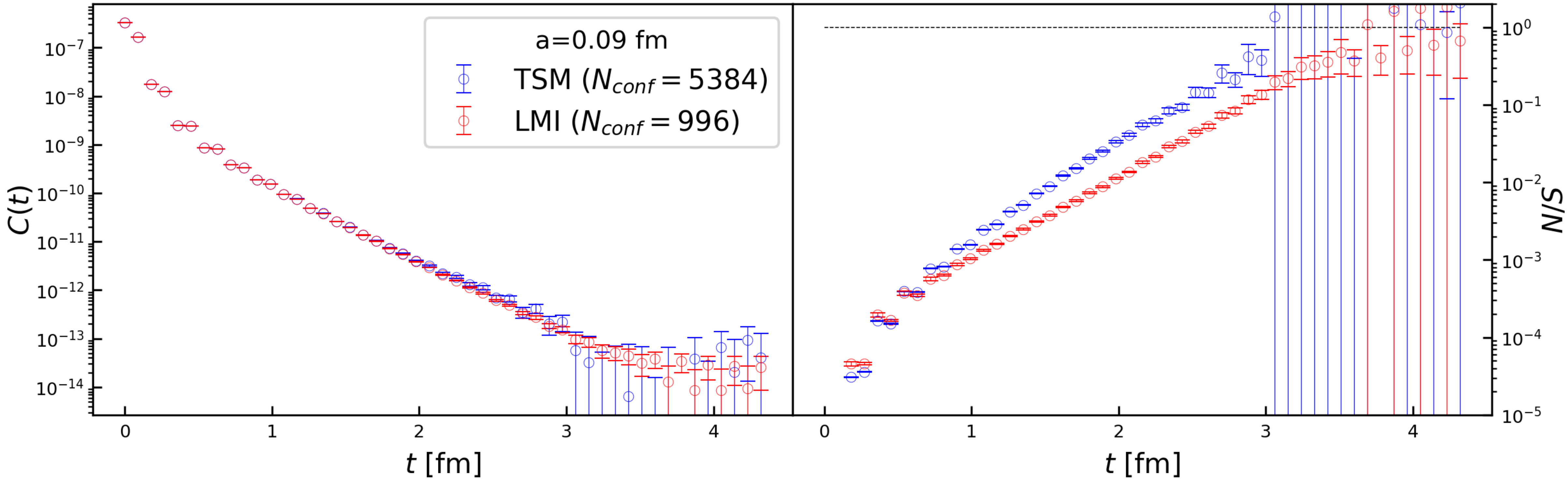
Application to connected g-2

- Calculated LMI local vector current at physical pion mass on three lattice spacings (0.12fm, 0.09fm, 0.06fm).
  - S. Lahert slides from Tues. 2:10pm.

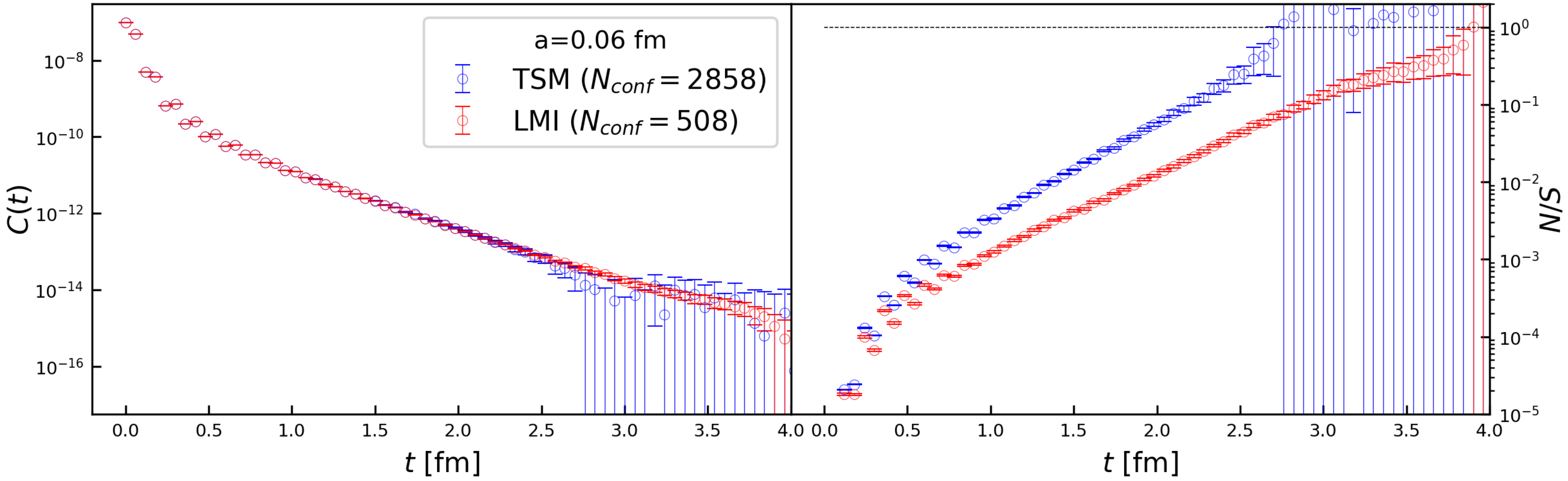
Application to connected g-2



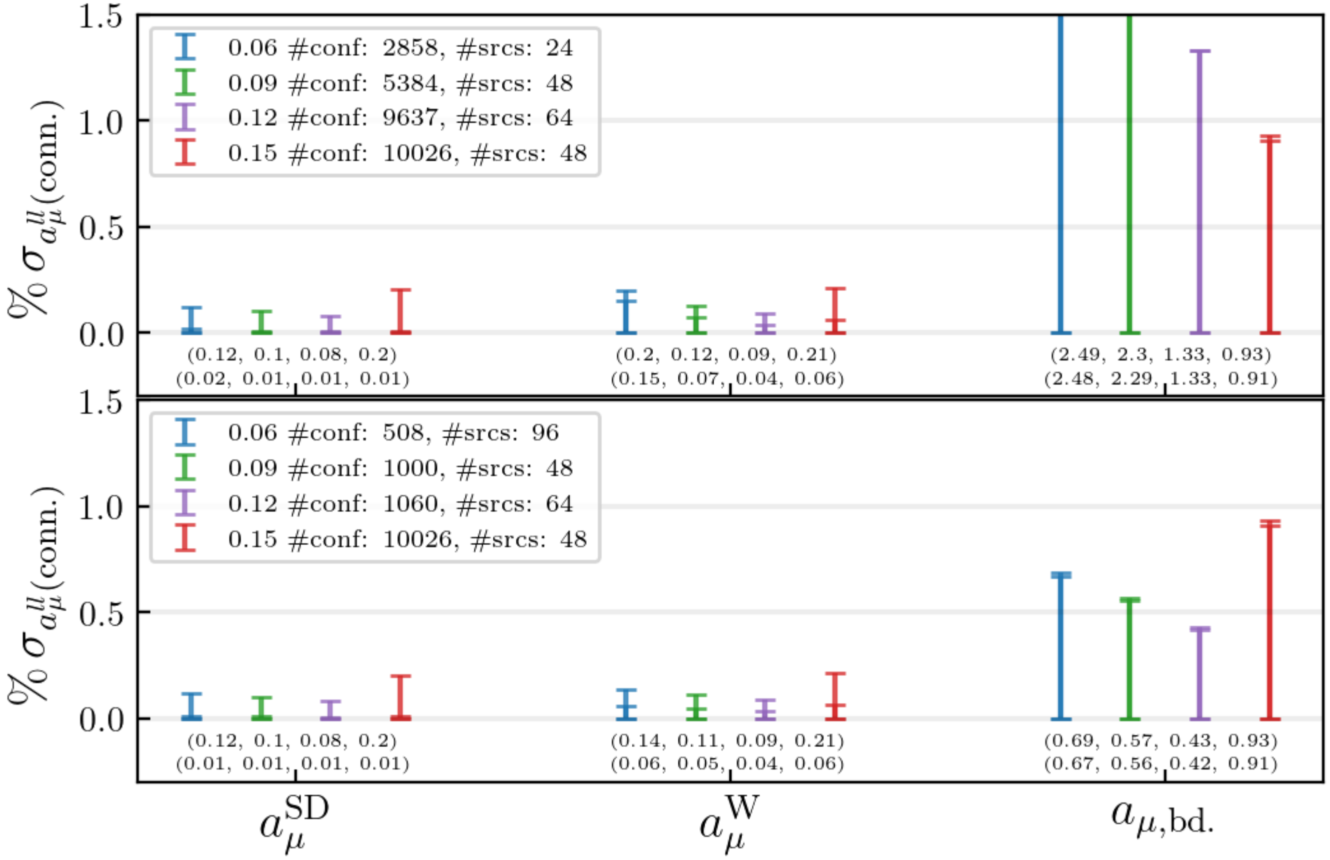
Application to connected g-2



Application to connected g-2



Application to connected g-2



Outlook

- Use All-to-all in three-point and four-point calculations.
  - Perturbative QED+QCD for g-2
  - Two pion scattering
- Performance tests for AMD MI250X ongoing.