MAX-PLANCK-INSTITUT
FÜR PLASMAPHYSIK

Reduced precision in GENE

Challenges, benefits & novel mixed-precision strategy

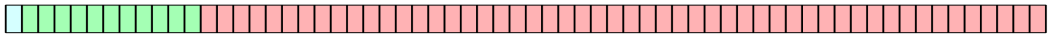
Carl-Martin Pfeiler (joint work with G. Merlo)

Max-Planck-Institut für Plasmaphysik

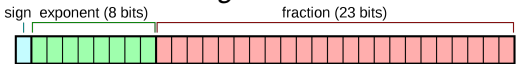
FP types, GPUs & MXP-HPL

Landscape of most important floating point types

IEEE - FP64: *double*



IEEE - FP32: *single*



BF16: *bfloat*

IEEE - FP16: *half*

Modern GPUs

- Natively support 16-bit types
- FP64 / FP32 performance stagnating
- Low precision tensor units

- FP64: $\varepsilon_{FP64} = 2^{-52} \approx 2 \cdot 10^{-16}$
- FP32: $\varepsilon_{FP32} = 2^{-23} \approx 2 \cdot 10^{-7}$
- BF16: $\varepsilon_{BF16} = 2^{-7} \approx 8 \cdot 10^{-3}$
- FP16: $\varepsilon_{FP16} = 2^{-10} \approx 1 \cdot 10^{-3}$, Range $\approx [1/16K, 65K]$

Mixed-precision performance potential

Rank	Computer	HPL-MxP (Eflop/s)	HPL Rmax (Eflop/s)	Speedup
hpl-mxp.org				
1	El Capitan	16.680	1.8090	9.2
2	Aurora	11.643	1.0120	11.5
3	Frontier	11.390	1.3530	8.4
4	JUPITER B	6.255	1.0000	6.3
5	CHIE-4	3.304	0.1354	24.4
6	ABCI 3.0	2.363[a]	0.1451	16.3
7	LUMI	2.350	0.3797	6.2
8	Fugaku	2.000	0.4420	4.5
9	Leonardo	1.842	0.2412	7.6

Mixed precision algorithm

- Intensive work in lower precision
- Cheap postprocessing to improve result accuracy
- Iterate until high precision recovered

Half precision port of GENE

Half precision software support

Half precision library / language support

- Fortran: none ⇝ tell it it's working with `C_INT16_T`
- MPI implementations: no ⇝ `MPI_INT16_T` or use nccl / rccl
- C++: optional support from C++23 on ⇝ but access to CUDA / HIP types
- cuFFT / hipFFT: prohibitive shortcomings ⇝ waiting for things to improve...

Extensions to GENE dependencies

- gtensor: New 16-bit types based on `cuda_fp16.h`, `cuda_bf16.h`
- reg_storage: Memory operations of appropriate (short) integer type

  ⇝ GENE: Can use half precision memory operations and slicing
  ⇝ GENE: Computations only in C++ / on GPU after data reinterpretation

Potential for single node performance improvements

- Reference run in single precision (FP32)
- Accumulated timings of all RK stages for 2000 time steps
- Resolution is 2 x 256 x 80 x 20 x 48 x 12 ($\approx$ 472M) on 8 x A40 GPU
- Half precision (BF16) employed for f, g, h, emfields, bar_chi, rhs, moments
- Half precision not employed for FFT

Region label	FP32 [s]	BF16 [s]	Speedup	Operation / Dependency
CalFRhs0 w/o NL	349	174	2.01	rhs = rhs(f, g, h, emfields)
CalFRhs0 w/ NL	791	614	1.29	rhs = rhs(...) & FFT
coll_scheme (1x)	90.1	44.2	2.04	rhs += coll(f)

- Accuracy sufficient for simplest single-species test case
- Observe instabilities for more interesting (two-species) test case

GENE inter-node scaling bottleneck

GENE weak scaling from [Germaschewski et al 2021]

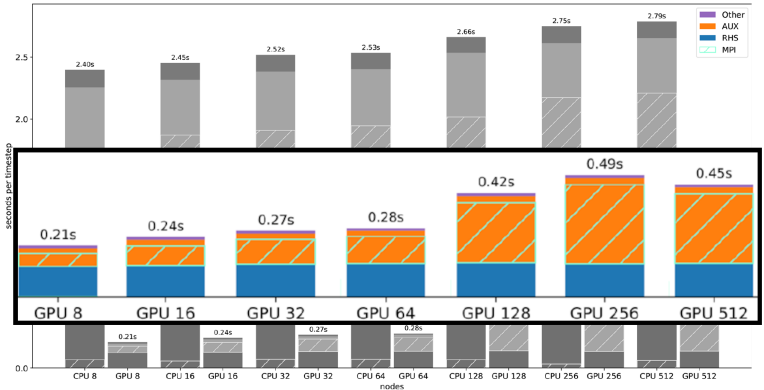
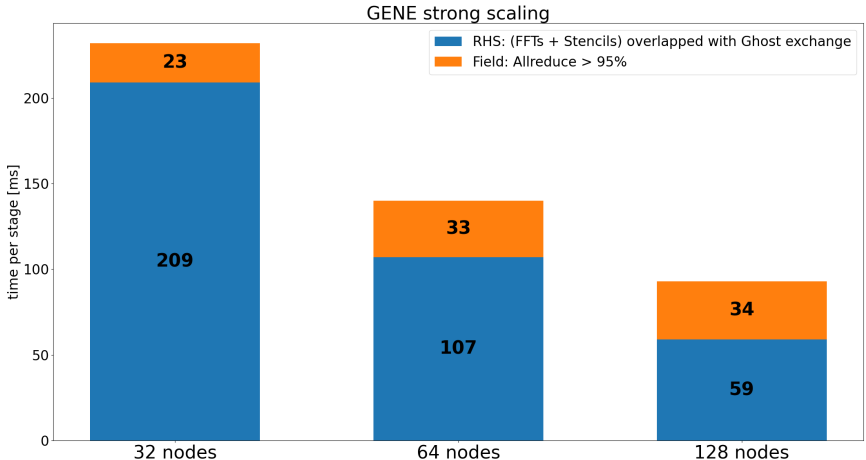


FIG. 7. Weak scaling to 512 Summit nodes, CPU vs GPU.

Communication bottleneck: `MPI_Allreduce` & Ghost exchanges

GENE strong scaling with optimized implementation

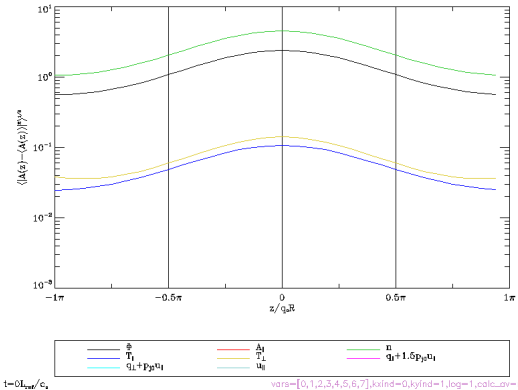


$(n_s, n_x, n_y, n_z, n_v, n_w) = (2, 1024, 512, 32, 48, 16)$, parallelization: $(2, 1, 1, 1, 2/4/8, 16)$

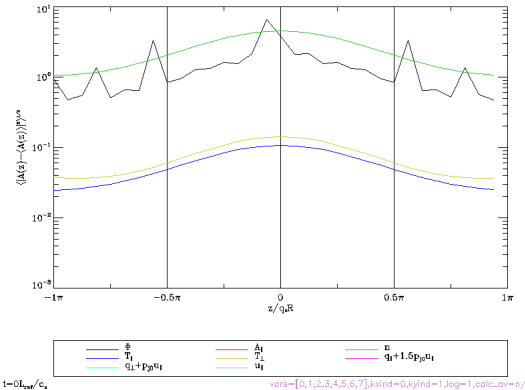
Applicability of half precision Allreduce

Low precision moments impact electromagnetic fields ($T = 0$)

FP32 moments



BF16 moments



$\rightsquigarrow$ Low precision moments Allreduce turns out to be problematic

Why is low precision moments Allreduce problematic?

- BF16 numbers are $\varepsilon_{\text{BF16}} = 1/128 < 1\%$ accurate
- Observe Φ more than 100% off — Why?

In local code Poisson equation simplifies to $\Phi = Diag^{-1} mom$

$$mom(x,y,z) = \sum_s pf(s) \cdot \underbrace{\sum_{v,w} mat(x,y,z,v,w,s) \cdot g(x,y,z,v,w,s)}_{mom_s^{vw}(x,y,z)}$$

By plasma's quasi-neutrality

$$mom_i^{vw}(x,y,z) \approx mom_e^{vw}(x,y,z) \quad \& \quad charge(s) = \pm 1$$

Taking the difference of two close-by values is numerically ill-conditioned!

Catastrophic cancellation

Data $x, y \in \mathbb{R}$ and floating point system $(\mathbb{F}, \varepsilon_{\mathbb{F}})$.

- Cast $\mathbb{F}(x) := \operatorname{argmin}\{|x - f| : f \in \mathbb{F}\}$

$$\implies \quad \text{Relative error:} \quad \frac{|x - \mathbb{F}(x)|}{|x|} \leq \frac{\varepsilon_{\mathbb{F}}}{2}$$

- Difference $y \ominus^{\mathbb{F}} x := \mathbb{F}(\mathbb{F}(y) - \mathbb{F}(x))$, where $x \sim^{\varepsilon} y$ are close

$$\implies \quad \text{Relative error:} \quad \frac{|(y - x) - (y \ominus^{\mathbb{F}} x)|}{|y - x|} \leq \frac{1}{2} \cdot \frac{\varepsilon_{\mathbb{F}}}{\varepsilon}, \qquad \text{where } \varepsilon := \frac{|y - x|}{|x| + |y|} \ll 1$$

- Example: Base-10 system with $\varepsilon_{\mathbb{F}} = 10^{-2}$, $x = 1.23456$, $y = 1.23554$, $\varepsilon \approx 10^{-3}$

$$y - x = 0.00098 \quad \text{but} \quad y \ominus^{\mathbb{F}} x = 1.24 - 1.23 = 0.01 \qquad \implies \quad \text{Rel. err.} \approx 900\%$$

Mixed-precision baseline reduction

Baseline reduction idea

$x \sim y$ implies existence of *baseline* $b \in \mathbb{R}$ s.t.

$$\implies \quad x = b + \delta_x \quad \& \quad y = b + \delta_y \qquad \text{with} \qquad \delta_x \not\sim \delta_y$$

$$\implies \quad y - x = \delta_y - \delta_x \quad \& \quad \delta_y -^{\mathbb{F}} \delta_x \quad \text{not problematic numerically}$$

Ex: Base-10 system with $\varepsilon_{\mathbb{F}} = 10^{-2}$, $x = 1.23456$, $y = 1.23554$, new $b = 1.2$

$$y - x = 0.00098\,, \quad \delta_x = 0.03456 = 3.456 \cdot 10^{-2} \quad \& \quad \delta_y = 0.03554 = 3.554 \cdot 10^{-2}$$

$$\delta_y -^{\mathbb{F}} \delta_x = (3.55 - 3.46) \cdot 10^{-2} = 0.09 \cdot 10^{-2} = 0.0009 \quad \implies \text{Rel. err.} \approx 8\%$$

Obvious questions:

- Baseline effect?
- Which baseline?
- Overhead baseline?

Theoretical baseline reduction effect

Close values

$$\varepsilon := \frac{|m_i - m_e|}{|m_i| + |m_e|} \ll 1 \quad \& \quad \varepsilon_b := \min\left\{ \frac{|m_i - b|}{|m_i| + |b|}, \frac{|m_e - b|}{|m_e| + |b|} \right\}$$

Normal precision-$\mathbb{L}$ reduction

$$\frac{\left|(m_i - m_e) - (m_i -^{\mathbb{L}} m_e)\right|}{|m_i - m_e|} \leq \frac{\varepsilon_{\mathbb{L}}}{\varepsilon}$$

Theoretical baseline reduction

$$\frac{\left|(m_i - m_e) - (m_i - b) -^{\mathbb{L}} (m_e - b))\right|}{|m_i - m_e|} \leq \left(1 + \frac{\varepsilon_b}{\varepsilon}\right) \cdot \varepsilon_{\mathbb{L}}$$

Practical baseline reduction effect

In practice, also $m_i - b$ has to be done in some finite precision

$$\frac{\left|(m_i - m_e) - (m_i -^{\mathbb{L}} b) -^{\mathbb{L}} (m_e -^{\mathbb{L}} b))\right|}{|m_i - m_e|} \leq \left(1 + \frac{\varepsilon_b}{\varepsilon} + \frac{1}{\varepsilon}\right) \cdot \varepsilon_{\mathbb{L}}$$

$(\mathbb{L}, \mathbb{H})$-*mixed precision baseline reduction* ($0 < \varepsilon_{\mathbb{H}} < \varepsilon_{\mathbb{L}} < 1$)

$$\frac{\left|(m_i - m_e) - (m_i -^{\mathbb{H}} b) -^{\mathbb{L}} (m_e -^{\mathbb{H}} b))\right|}{|m_i - m_e|} \leq \frac{\varepsilon_{\mathbb{H}}}{\varepsilon} + \left(1 + \frac{\varepsilon_b}{\varepsilon}\right) \cdot \varepsilon_{\mathbb{L}}$$

E.g., $(\mathbb{L}, \mathbb{H})$ = (BF16, FP32)

Pointwise analysis transfers to global norm

Normal $\mathbb{F}$-reduction

$$\frac{\left\|(\boldsymbol{m}_i - \boldsymbol{m}_e) - (\boldsymbol{m}_i -^{\mathbb{L}} \boldsymbol{m}_e)\right\|}{\|\boldsymbol{m}_i - \boldsymbol{m}_e\|} \leq \left(1 + \frac{\|\boldsymbol{m}_i\| + \|\boldsymbol{m}_e\|}{\|\boldsymbol{m}_i - \boldsymbol{m}_e\|}\right) \cdot \frac{\varepsilon_{\mathbb{L}}}{2}$$

$(\mathbb{L}, \mathbb{H})$-mixed precision baseline reduction

$$\begin{aligned}&\frac{\left\|(\boldsymbol{m}_i - \boldsymbol{m}_e) - \left((\boldsymbol{m}_i -^{\mathbb{H}} \boldsymbol{b}) -^{\mathbb{L}} (\boldsymbol{m}_e -^{\mathbb{H}} \boldsymbol{b})\right)\right\|}{\|\boldsymbol{m}_i - \boldsymbol{m}_e\|} \\ &\quad\leq \left(1 + \frac{\|\boldsymbol{m}_i\| + \|\boldsymbol{m}_e\|}{\|\boldsymbol{m}_i - \boldsymbol{m}_e\|}\right) \cdot \frac{\varepsilon_{\mathbb{H}}}{2} + \left(1 + \frac{\min\left\{\|\boldsymbol{m}_i - \boldsymbol{b}\|, \|\boldsymbol{m}_e - \boldsymbol{b}\|\right\}}{\|\boldsymbol{m}_i - \boldsymbol{m}_e\|}\right) \cdot \varepsilon_{\mathbb{L}}\end{aligned}$$

Moreover, the relative error's expected value $\approx \frac{1}{\sqrt{12}} \times$ worst-case estimate

How to obtain a good baseline?

Analysis implies

- Baseline shall approximate $\boldsymbol{m}_i$ or $\boldsymbol{m}_e$
- The better the baseline, the better the result
- Aim for $\varepsilon_{\boldsymbol{b}} \leq \max\{\varepsilon_{\mathbb{H}}/\varepsilon_{\mathbb{L}}, \varepsilon\}$ at least
- No significant improvements for $\varepsilon_{\boldsymbol{b}} \ll \max\{\varepsilon_{\mathbb{H}}/\varepsilon_{\mathbb{L}}, \varepsilon\}$

Suggestions for baseline choice

- Analytical *a priori* knowledge, e.g., $data(t; w) = exp(-|w|) + pert(t)$
- For time-marching scheme: (high-order) extrapolation, e.g.,

$$data(t) \approx data(t - \Delta t) =: \boldsymbol{b} \quad \Longrightarrow \quad \|\varepsilon_{\boldsymbol{b}}\| = \mathcal{O}(\Delta t)$$

$$data(t) \approx 2 \cdot data(t - \Delta t) - data(t - 2\Delta t) =: \boldsymbol{b} \quad \Longrightarrow \quad \|\varepsilon_{\boldsymbol{b}}\| = \mathcal{O}\left(\Delta t^2\right)$$

Mixed precision baseline reduction in time-marching scheme: Overhead?

Postal model for send / receive cost

$$T_{s./r.} = \alpha + n \cdot \beta$$

with start-up time α per message, and inverse bandwidth β, for n bytes.

Allreduce cost for n bytes per each of p processes [Rabenseifner 2004]

$$T_{all,h.\&d.} = (3 + 2\log_2 p) \cdot \alpha + 4n\beta$$
$$T_{all,ring} = 2(p-1) \cdot \alpha + 2n\beta$$

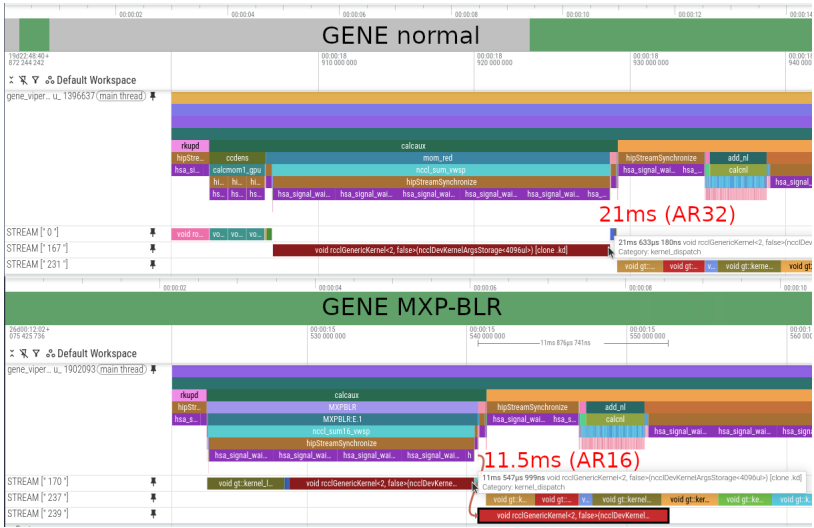
FP32 Allreduce vs. FP32/BF16-MXP-Baseline Allreduce, $N = 2 \cdot n_x n_y n_z \cdot nfields$

$$T_{all,FP32,ring} = 2(p-1) \cdot \alpha + 2 \cdot sizeof(FP32) \cdot N \cdot \beta$$
$$T_{all,MXP,ring} = 2(p-1) \cdot \alpha + 2 \cdot sizeof(BF16) \cdot N \cdot \beta + \alpha + sizeof(FP32) \cdot N \cdot \beta$$

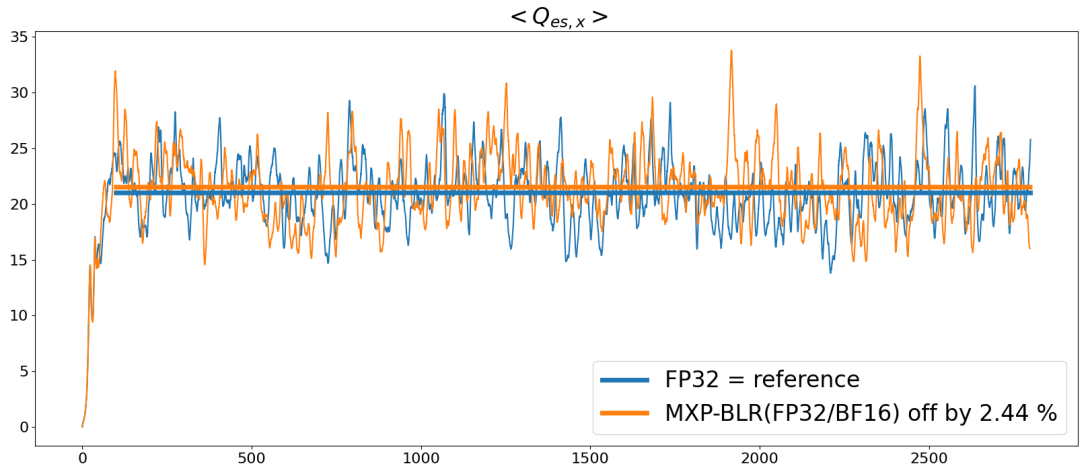
Moments calculation with baseline & CUDA streams

- `StreamWaitEvent(bcast_event)` ...GPU-main_stream
- `lmom = calc_local_moments(g)` ...GPU-main_stream
- `mom = pf * (lmom - baseline)` ...GPU-main_stream
- `RecordEvent(main_event)` ...GPU-main_stream
- `StreamWaitEvent(main_event)` ...GPU-bcast_stream
- `Allreduce16(mom, comm=comm_vwspec)` ...GPU-main_stream
- `Broadcast(src=lmom, tgt=baseline, comm=comm_spec)` ...GPU-bcast_stream
- `RecordEvent(bcast_event)` ...GPU-bcast_stream

Faster Allreduce & baseline communication overhead hidden



Two species cyclone base case simulated accurately



Summary & Outlook

Summary & Outlook

Half precision port

- Running GENE in half precision technically cumbersome, but possible
- Expected speedups realized
- Accuracy not satisfactory beyond simple test case

Novel mixed-precision baseline reduction

- Motivated by Allreduce scaling bottleneck
- Applicable beyond simple test case
- Formulated, analyzed, implemented, potential demonstrated
- Design of large scale demonstration setup ongoing