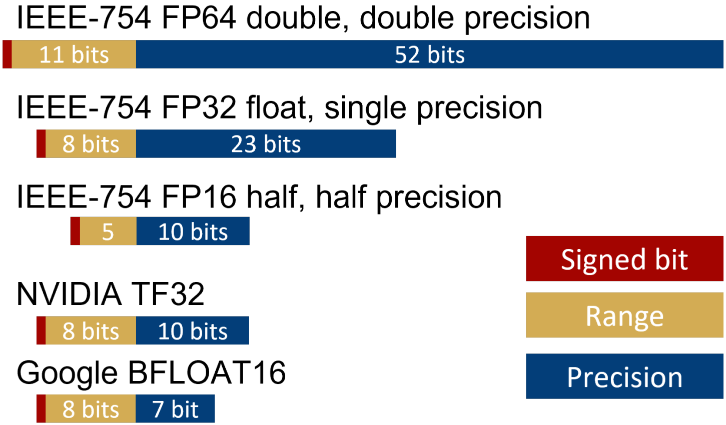
# Mixed precision: when is it worth it?

**Bálint Siklósi**, István Reguly

Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

November 10, 2021

# Floating point representation



# Key challenges of exascale computing

In 2008 Exascale Study Group (ESG) issued a report: Technology Challenges in Achieving Exascale Systems [8]

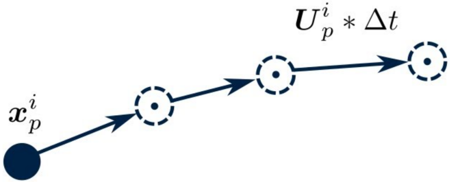
1. power consumption
   - 600 MW $\rightarrow$ 20 MW
2. speed and energy of data movement
   - time of data movement $>$ time of FLOP
3. fault tolerance
   - Failures happen faster, than checkpointing a job.
4. extreme parallelism
   - To compute at a rate of 1 exaflop requires 1 billion floating point units performing 1 billion calculations per second each

# Wide interest in mixed precision

- Less communication
  - Reduce memory traffic
  - Reduce network traffic
- Reduce memory footprint
- More Flop per second
  - Reduced energy consumption
  - Reduced time to compute

- Gordon Bell Prize winner climate simulation [9]
- Gordon Bell Prize winner opioid addiction research [7]
- Best paper at ISC'19 GPUMixer [10]
- Earthquake simulation [6]
- Mixed precision in-memory computing [11]
- AI, Deep learning [13], [12]
- Linear solvers, numerical methods [1], [2], [4]

# Particle method




Position: FP64
Displacement: FP32

# Machine learning [12]

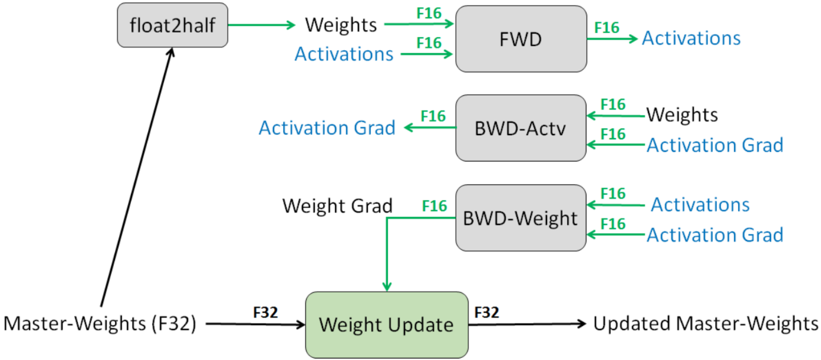



Figure: Mixed precision training iteration for a layer. Memory consumption of deep learning models nearly halved.

# Linear solver in Ginkgo[3]

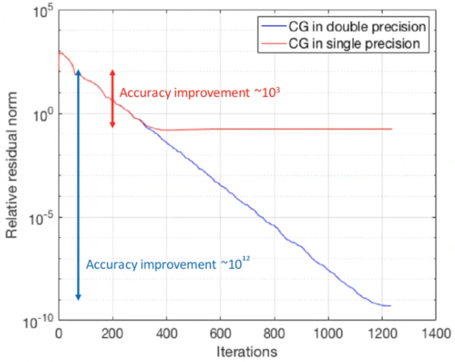
Linear system Ax=b
Rule of thumb [5]:

$$\text{relative residual accuracy} = (\text{unit round-off}) * (\text{linear system's condition number})$$

# Linear solver in Ginkgo[3]

Linear system Ax=b with cond(A)=4
Rule of thumb [5]:

relative residual accuracy = (unit round-off) $*$ (linear system's condition number)

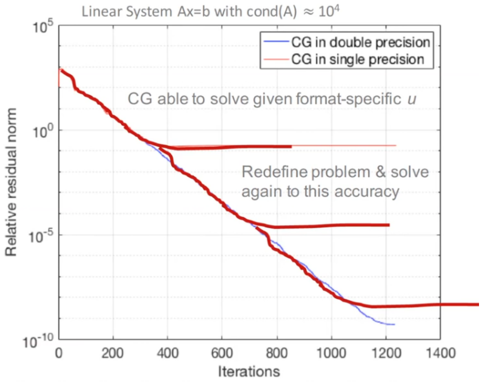


# Linear solver in Ginkgo[3]

Linear system Ax=b with cond(A)=4  16% runtime improvement

Rule of thumb [5]:

$$\text{relative residual accuracy} = (\text{unit round-off}) * (\text{linear system's condition number})$$



# GPUMixer [10]

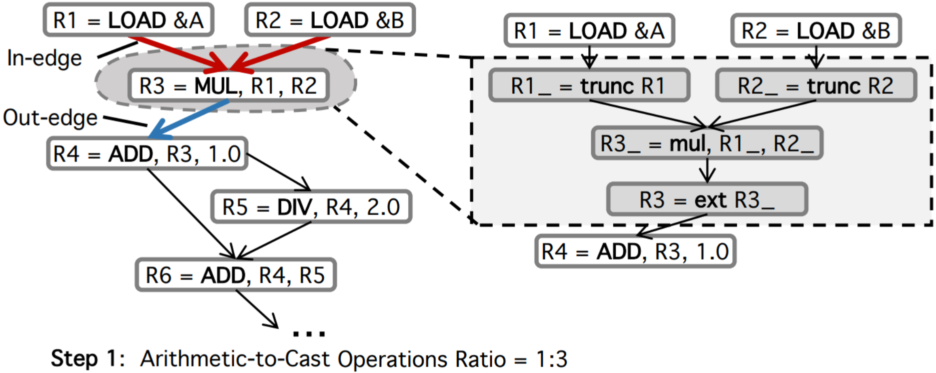



Figure: Illustration of the algorithm to find Fast Imprecise Sets (FISets)

# GPUMixer [10]

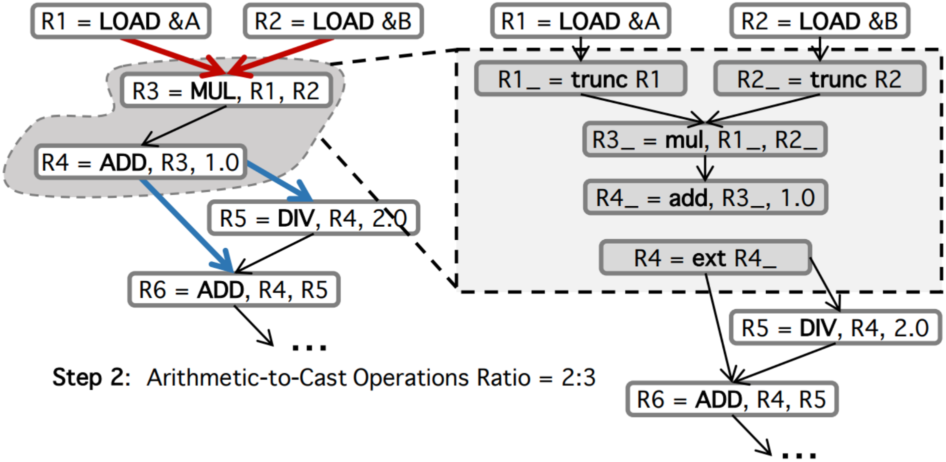



Figure: Illustration of the algorithm to find Fast Imprecise Sets (FISets)

# GPUMixer [10]

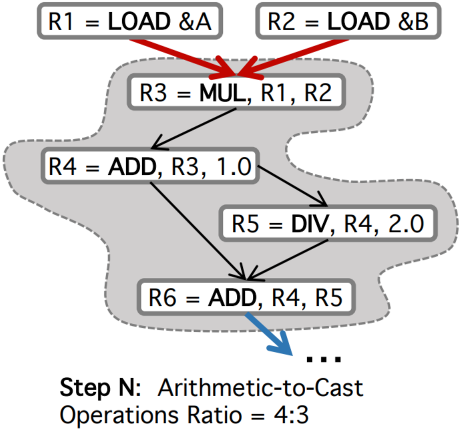
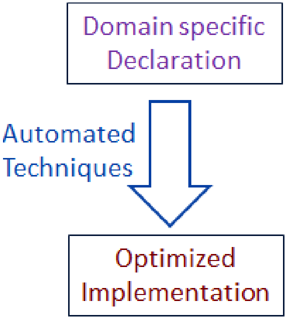



Figure: Illustration of the algorithm to find Fast Imprecise Sets (FISets) 46.4% of the ideal speedup

One approach to develop future proof HPC applications is the use of domain specific high-level abstractions (HLAs)

- Provide the application developer with a domain specific abstraction
  - To declare the problem to be computed
  - Without specifying its implementation
  - Use domain specific constructs in the declaration
- Create a lower implementation level
  - To apply automated techniques for translating the specification to different implementations
  - Target different hardware and software platforms
  - Exploit domain knowledge for better optimisations on each hardware system

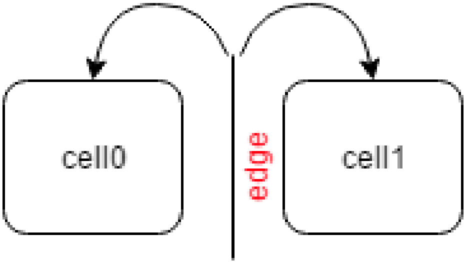


# OP2

- Open Source project
- OP2 based on OPlus (**O**xford **P**arallel **L**ibrary for **U**nstructured **S**olvers), developed for CFD codes on distributed memory clusters
- Separate high level description from parallel implementation
- Looks like a conventional library, but uses code transformations (source to source translator) to generate parallel codes
- Support application codes written in C++ or FORTRAN

## OP2 loop over edges

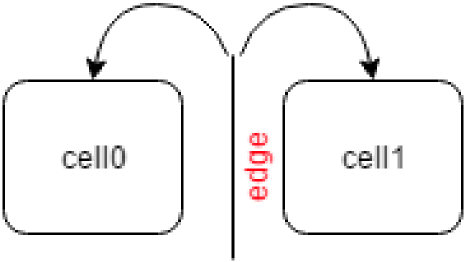
```
void res(double* edge,
         double* cell0,
         double* cell1){
   *cell0 += *edge;
   *cell1 += *edge;
}
```




```
op_par_loop(res,"residual_calculation", edges,
  op_arg(dedges, -1, OP_ID, 1, "double", %OP_READ),
  op_arg(dcells, 0, pecell, 1, "double", OP_INC),
  op_arg(dcells, 1, pecell, 1, "double", OP_INC));
```

## OP2 loop over edges

```
void res(double* edge,
        float* cell0,
        float* cell1){
   *cell0 += *edge;
   *cell1 += *edge;
}
```




```
op_par_loop(res,"residual_calculation", edges,
  op_arg(dedges, -1, OP_ID, 1, "double", %OP_READ),
  op_arg(dcells, 0, pecell, 1, "float", OP_INC),
  op_arg(dcells, 1, pecell, 1, "float", OP_INC));
```

# Example test

- CPU related environment
  - Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz with 64 processes
- GPU related environment
  - dgx-station with 2 Nvidia V100 GPUs

- Test application
  - **Airfoil**, a standard finite volume CFD benchmark code with a 2.8M mesh size

| Environment | FP64 | FP32 | Mixed | speedup - mixed | speedup - float |
|---|---|---|---|---|---|
| Intel | 14.44s | 8.22s | 12.76s | 1.13 | 1.76 |
| Nvidia | 2.17s | 1.51s | 1.97s | 1.1 | 1.44 |

# Contact

- **OP-DSLs**: `https://op-dsl.github.io/`
- **OP2**: `https://github.com/OP-DSL/OP2-Common`
- siklosi.balint@itk.ppke.hu

Új Nemzeti Kiválóság Program

INNOVÁCIÓS ÉS TECHNOLÓGIAI MINISZTÉRIUM

SUPPORTED BY THE ÚNKP-21-3 NEW NATIONAL EXCELLENCE PROGRAM OF THE MINISTRY FOR INNOVATION AND TECHNOLOGY FROM THE SOURCE OF THE NATIONAL RESEARCH, DEVELOPMENT AND INNOVATION FUND.

[1] Ahmad Abdelfattah et al. "A survey of numerical linear algebra methods utilizing mixed-precision arithmetic". In: *The International Journal of High Performance Computing Applications* 35.4 (2021), pp. 344–369. DOI: 10.1177/10943420211003313. eprint: https://doi.org/10.1177/10943420211003313. URL: https://doi.org/10.1177/10943420211003313.

[2] Ahmad Abdelfattah et al. *A Survey of Numerical Methods Utilizing Mixed Precision Arithmetic*. 2020. arXiv: 2007.06674 [cs.MS].

[3] Hartwig Anzt et al. "Ginkgo: A high performance numerical linear algebra library". In: *Journal of Open Source Software* 5.52 (2020), p. 2260. DOI: 10.21105/joss.02260. URL: https://doi.org/10.21105/joss.02260.

[4] Azzam Haidar et al. "Mixed-Precision Iterative Refinement using Tensor Cores on GPUs to Accelerate Solution of Linear Systems". In: *Proceedings of the Royal Society A* 476 (2020-11 2020). ISSN: 1471-2946. DOI: https://doi.org/10.1098/rspa.2020.0110.

[5] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. 2002. DOI: 10.1137/1.9780898718027.

[6] Tsuyoshi Ichimura et al. "A Fast Scalable Implicit Solver for Nonlinear Time-Evolution Earthquake City Problem on Low-Ordered Unstructured Finite Elements with Artificial Intelligence and Transprecision Computing". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. SC '18. Dallas, Texas: IEEE Press, 2018.

[7] Wayne Joubert et al. "Attacking the Opioid Epidemic: Determining the Epistatic and Pleiotropic Genetic Architectures for Chronic Pain and Opioid Addiction". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. SC '18. Dallas, Texas: IEEE Press, 2018.

[8] Peter Kogge et al. "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems". In: *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Techinal Representative* 15 (Jan. 2008).

[9] Thorsten Kurth et al. "Exascale Deep Learning for Climate Analytics". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. SC '18. Dallas, Texas: IEEE Press, 2018. DOI: 10.1109/SC.2018.00054. URL: https://doi.org/10.1109/SC.2018.00054.

[10] Ignacio Laguna et al. "GPUMixer: Performance-Driven Floating-Point Tuning for GPU Scientific Applications". In: May 2019, pp. 227–246. ISBN: 978-3-030-20655-0. DOI: 10.1007/978-3-030-20656-7_12.

[11] Manuel Le Gallo et al. "Mixed-precision in-memory computing". In: *Nature Electronics* 1.4 (Apr. 2018), pp. 246–253. ISSN: 2520-1131. DOI: 10.1038/s41928-018-0054-8. URL: http://dx.doi.org/10.1038/s41928-018-0054-8.

[12] Paulius Micikevicius et al. *Mixed Precision Training*. 2018. arXiv: 1710.03740 [cs.AI].

[13] S. R. Nandakumar et al. "Mixed-Precision Deep Learning Based on Computational Memory". In: *Frontiers in Neuroscience* 14 (2020), p. 406. ISSN: 1662-453X. DOI: 10.3389/fnins.2020.00406. URL: https://www.frontiersin.org/article/10.3389/fnins.2020.00406.