Radio-astronomical imaging with HIP

Stefano Corda

stefano.corda@epfl.ch

04-10-2022

Swiss SKA Days, Lugano

In collaboration with:

Bram Veenboer

ASTRON

SKAO EPFL SCITAS SKACH

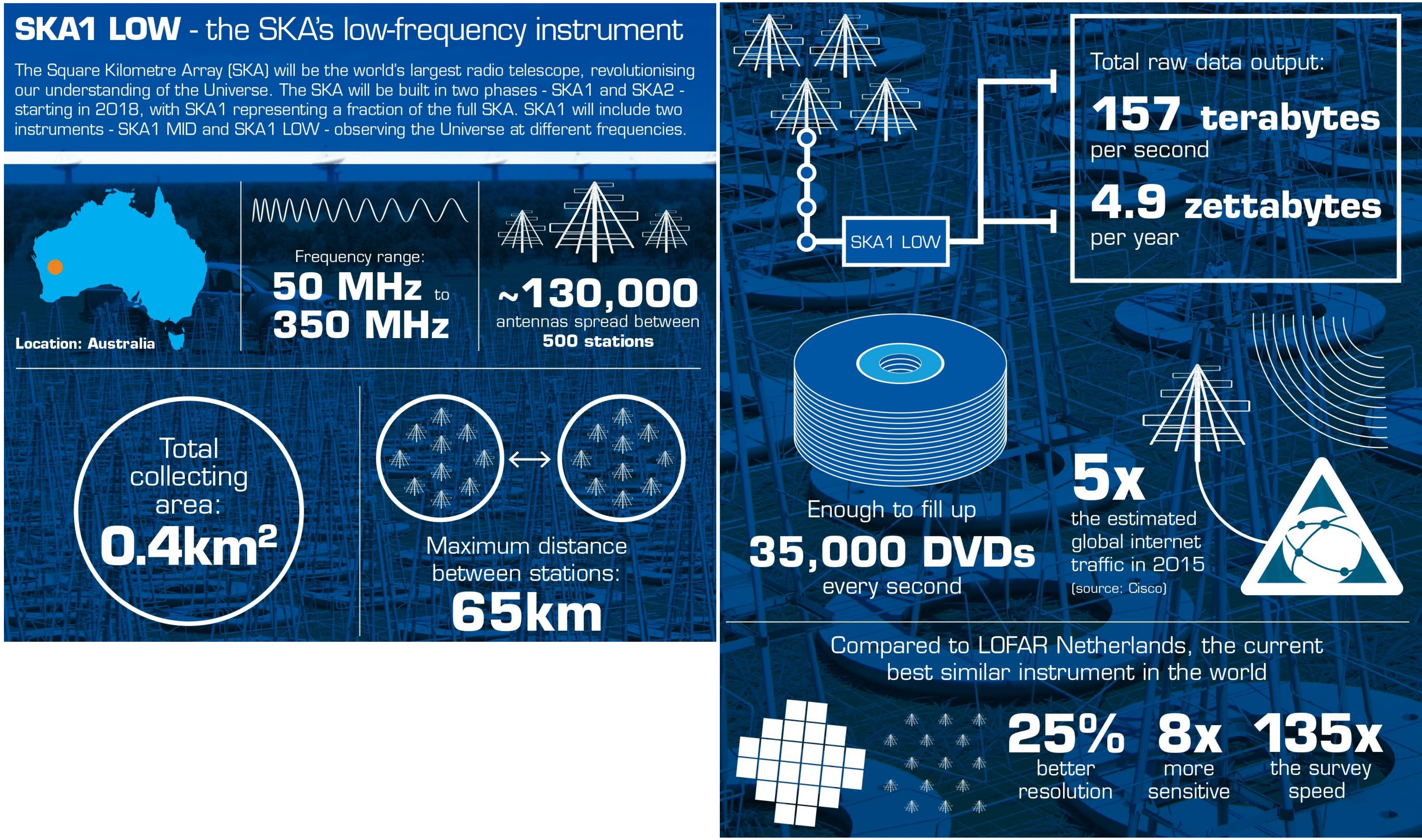
INFORMATIQUE SCIENTIFIQUE & SUPPORT APPLICATIF

SCIENTIFIC IT AND APPLICATION SUPPORT

Square Kilometre Array Challenge

SKA (Square Kilometre Array) requirements (per Science Data Processor (SDP) site):

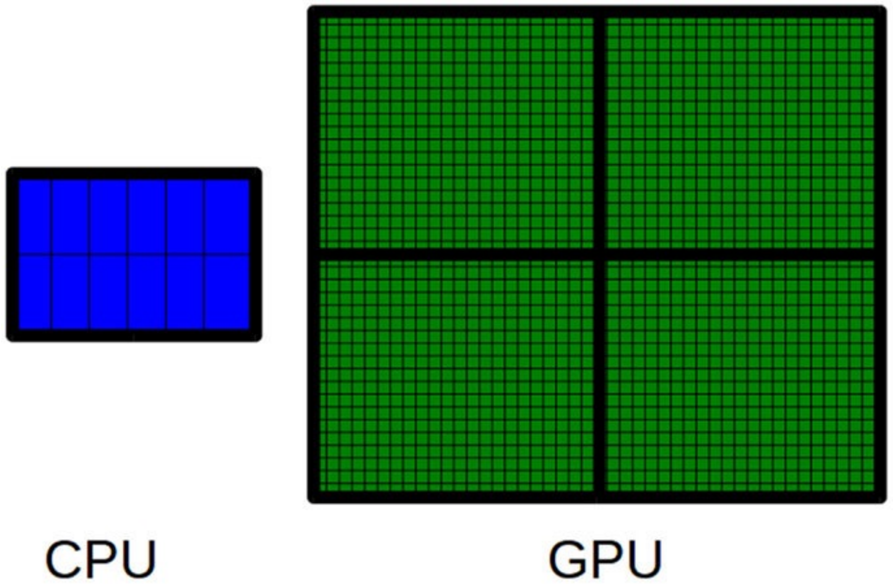
- ~ 260 PFlop/s
- ~ 157 TB/s
- ~ 5 MW



https://netherlands.skatelescope.org/2016/11/03/ogen-gericht-op-de-toekomst-tijdens-staatsbezoek-australie/

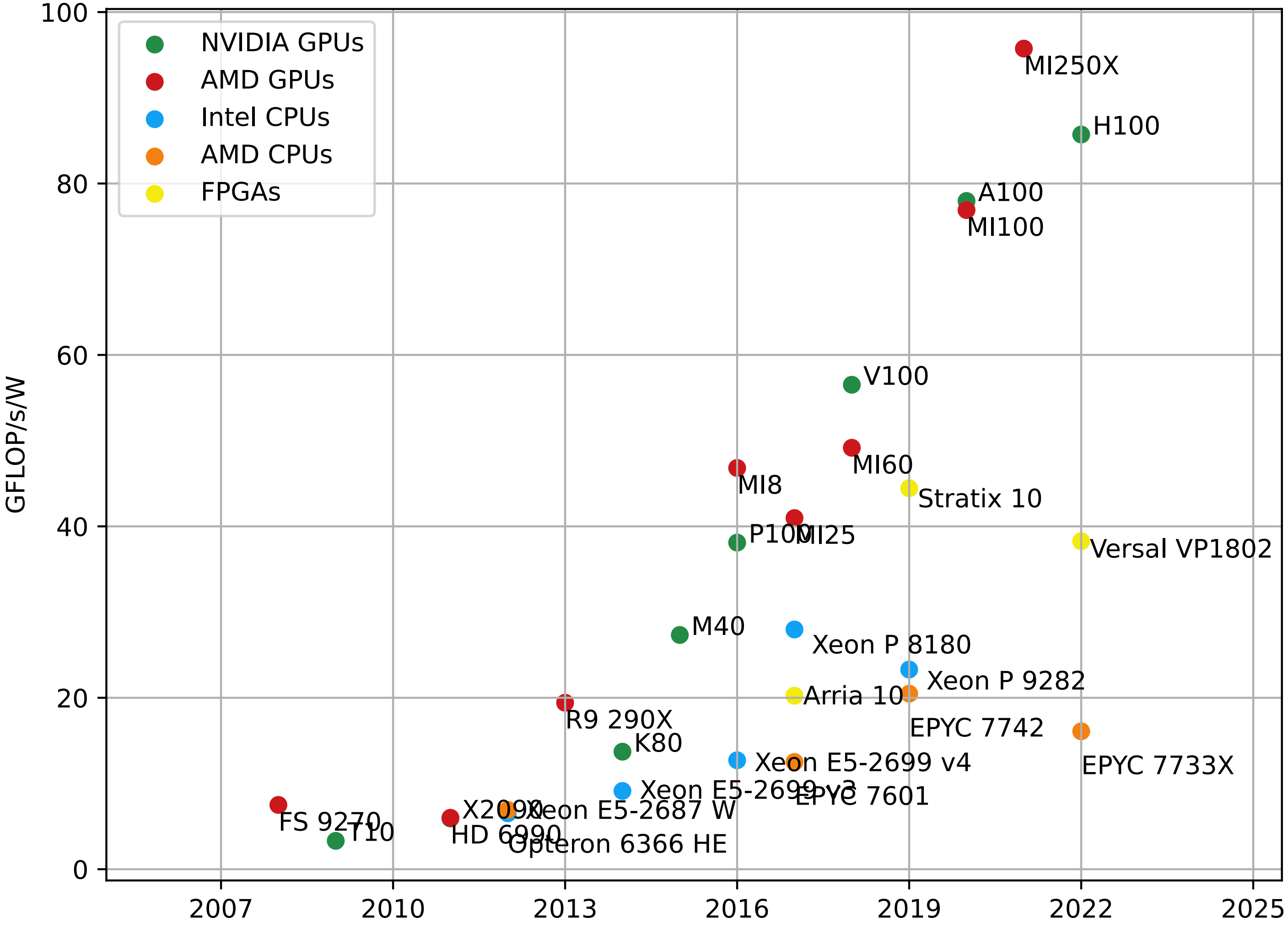
B. Veenboer et al., "Image-Domain Gridding on Graphics Processors" IPDPS 2017

Hardware trend

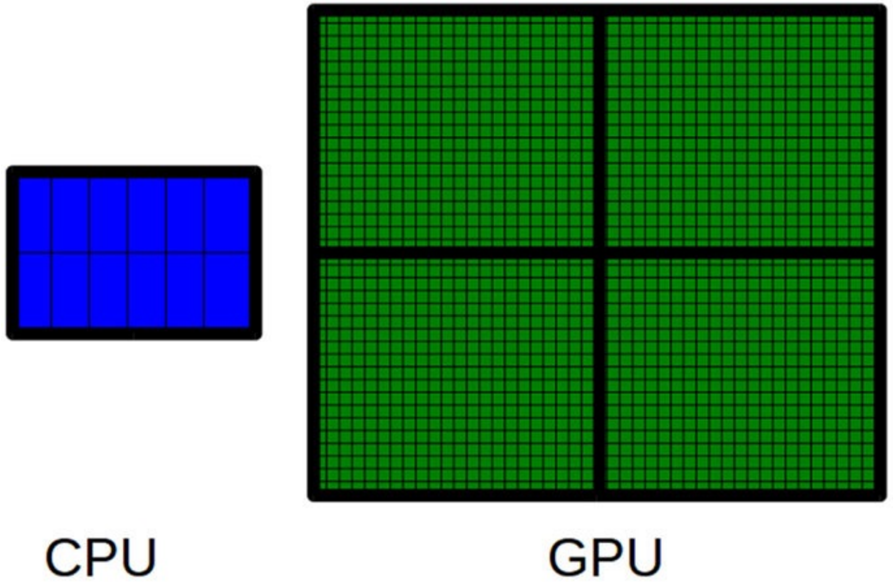


A. Manconi et al. “G-CNV: A GPU-based tool for preparing data to detect CNVs with read-depth methods”, Frontiers in Bioengineering and Biotechnology 2015

Hardware trend

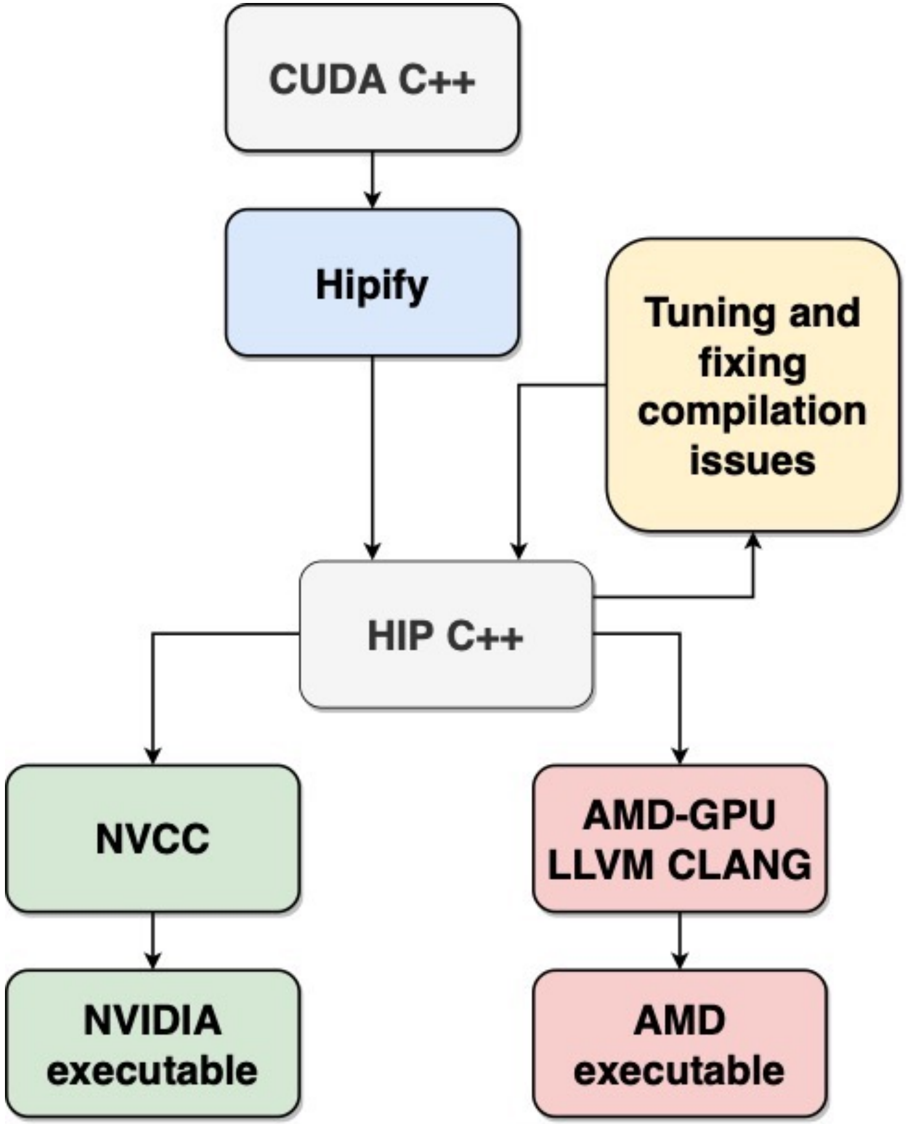


- Accelerators (GPUs and FPGAs) achieve better performance and energy efficiency than CPUs.
- Interesting competition in the GPU market (AMD vs NVIDIA).
- AMD systems are in the first positions of TOP500 and Green500 (ISC 22).



A. Manconi et al. "G-CNV: A GPU-based tool for preparing data to detect CNVs with read-depth methods", Frontiers in Bioengineering and Biotechnology 2015

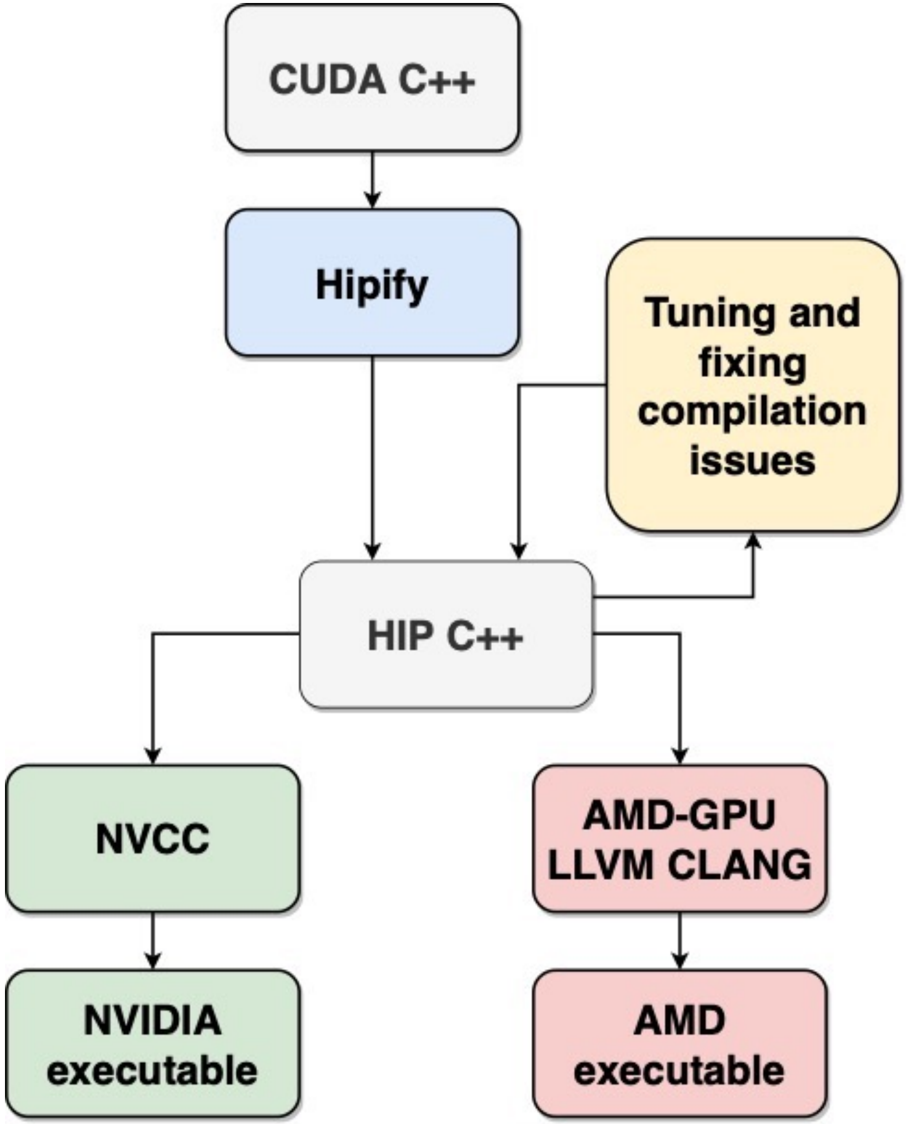
HIP



CUDA C++
Hipify
Tuning and fixing compilation issues
HIP C++
NVCC
AMD-GPU LLVM CLANG
NVIDIA executable
AMD executable

HIP

Portability: code is compiled with NVIDIA or AMD compilers.

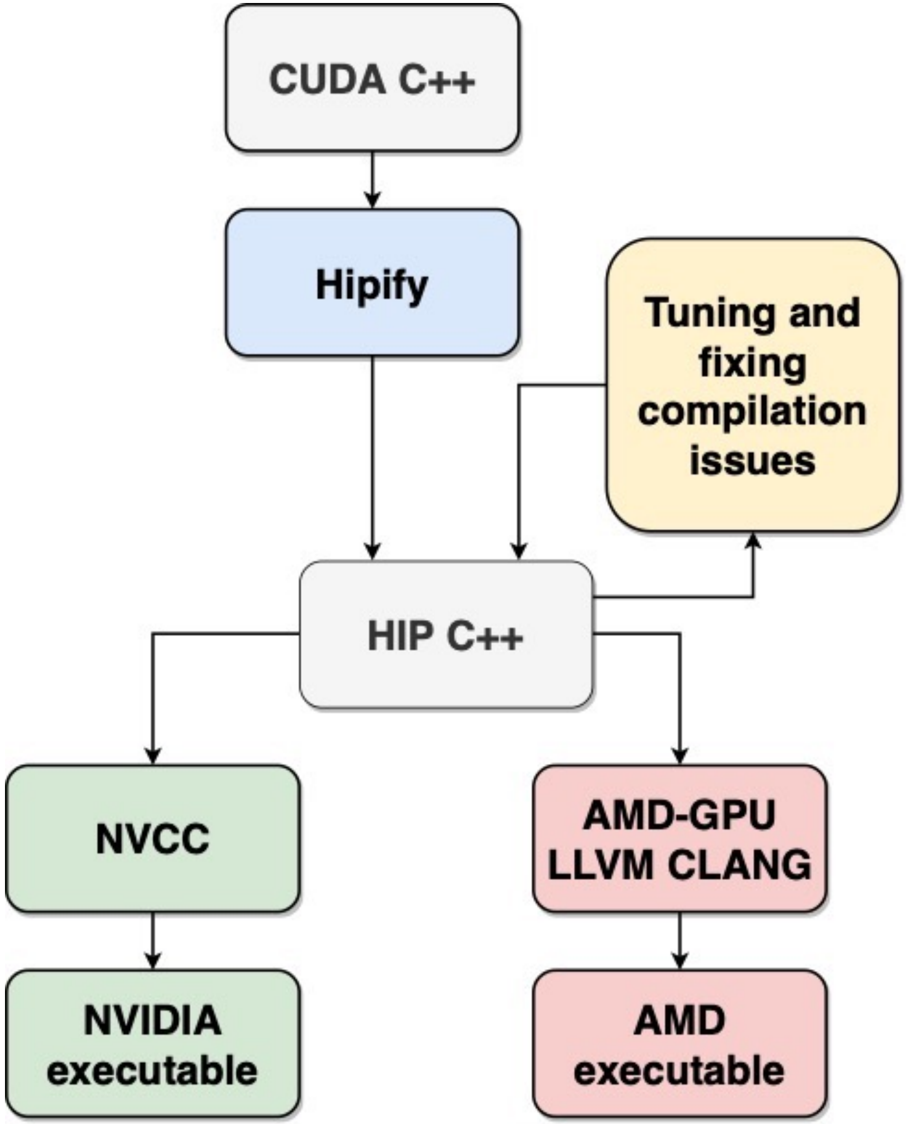


HIP

Portability: code is compiled with NVIDIA or AMD compilers.

Programmability: semantic is similar to CUDA (> 99%).

- First attempt (industry partner from France):
- Porting *all* CUDA code to HIP with Hipify
- Fixed some compilation issues
- Same performance on NVIDIA GPUs
- Poor performance on AMD GPUs



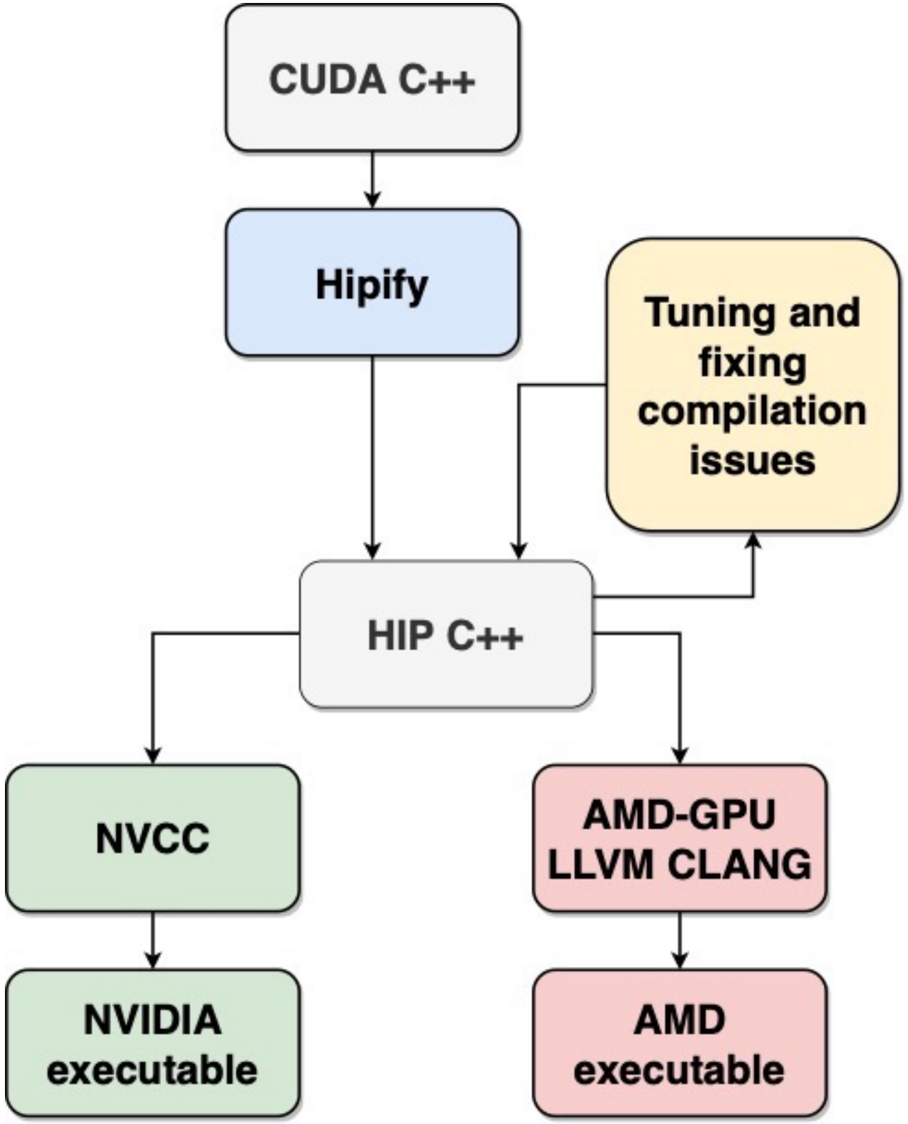
HIP

Portability: code is compiled with NVIDIA or AMD compilers.

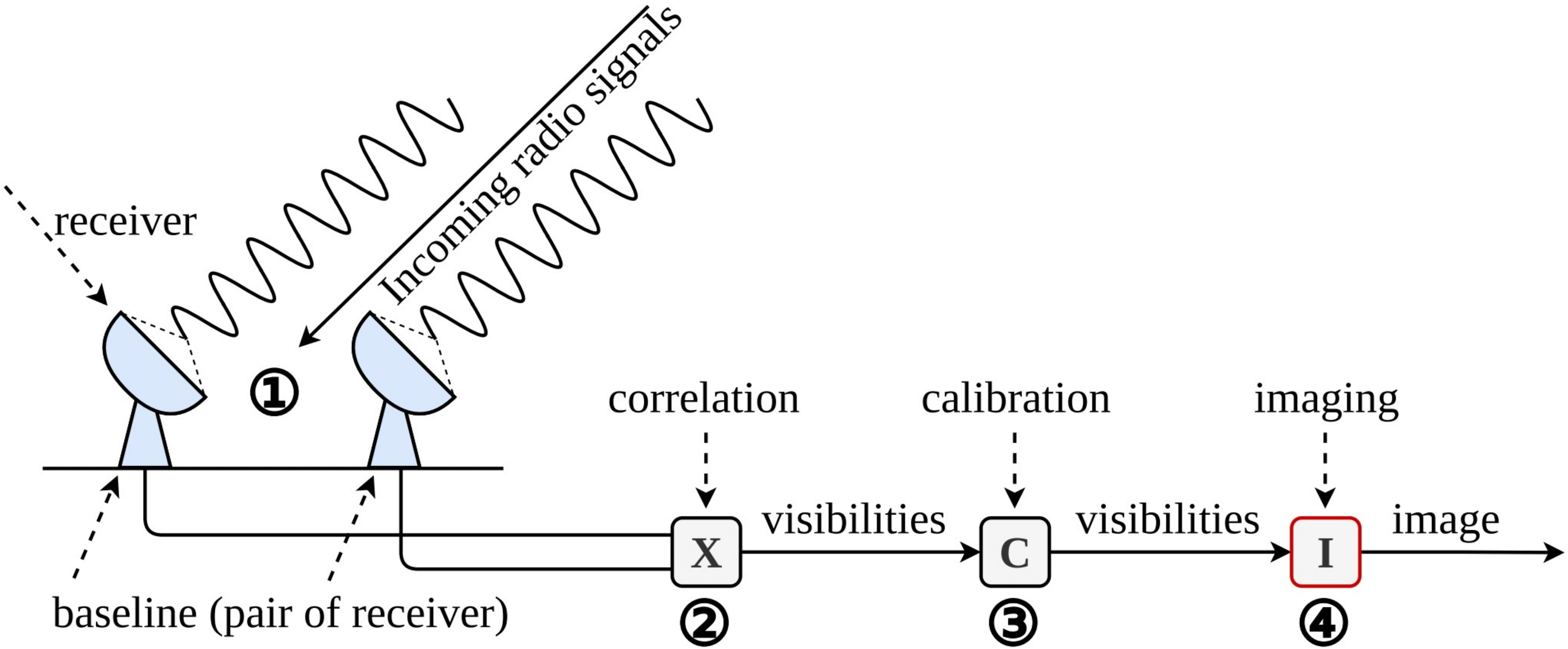
Programmability: semantic is similar to CUDA (> 99%).

- First attempt (industry partner from France):
- Porting *all* CUDA code to HIP with Hipify
- Fixed some compilation issues
- Same performance on NVIDIA GPUs
- Poor performance on AMD GPUs

→Start again, port one kernel at a time

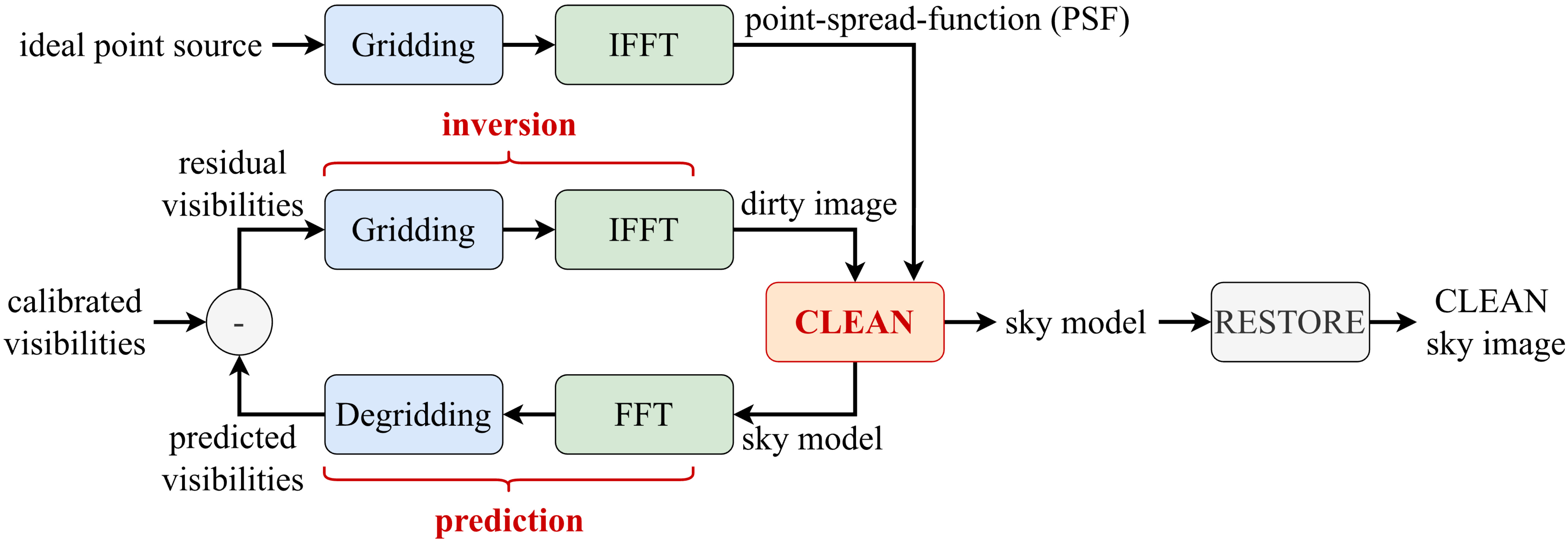


Interferometry



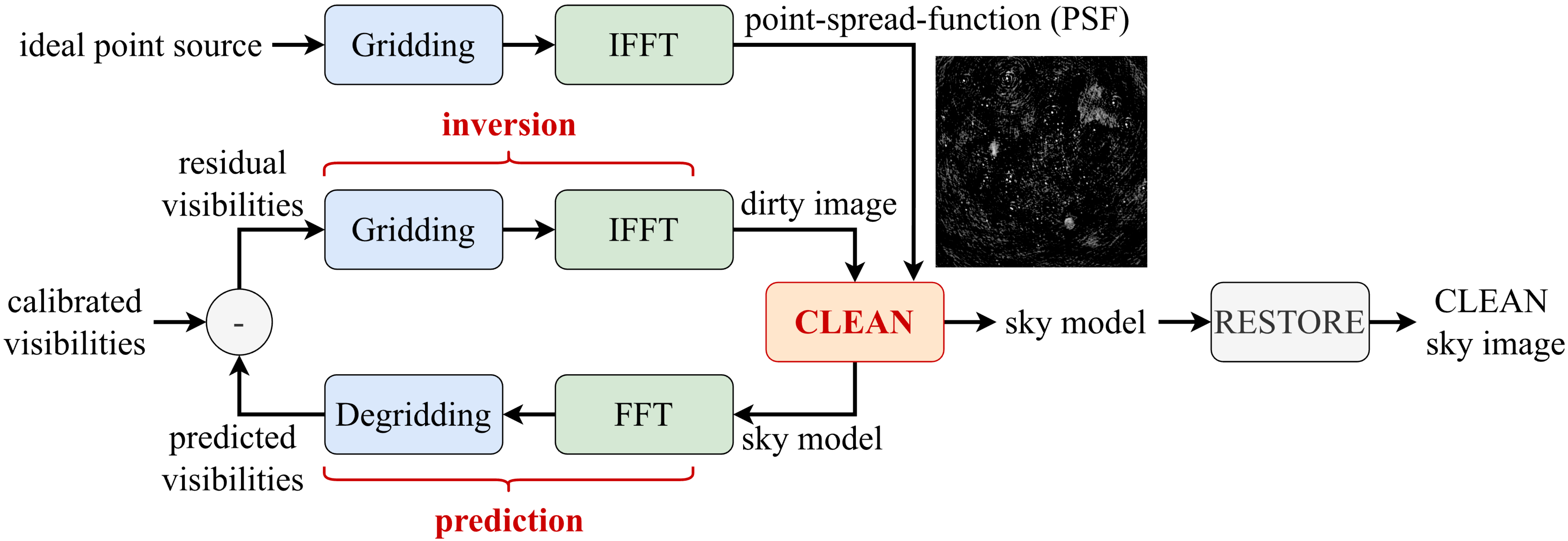
U. Rau et al., "Advances in Calibration and Imaging Techniques in Radio Interferometry", Proceedings of the IEEE 2009

Radio-astronomical imaging



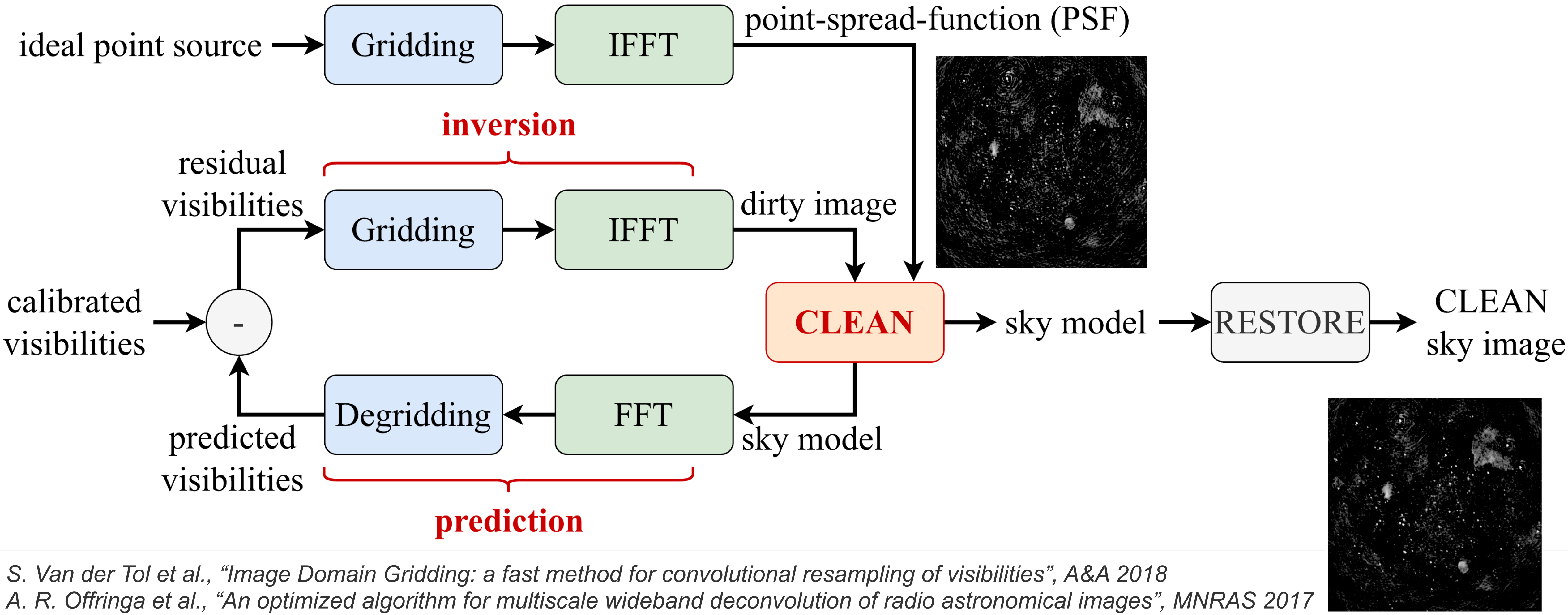
S. Van der Tol et al., "Image Domain Gridding: a fast method for convolutional resampling of visibilities", A&A 2018
A. R. Offringa et al., "An optimized algorithm for multiscale wideband deconvolution of radio astronomical images", MNRAS 2017
S. Corda et al, "Near memory acceleration on high resolution radio astronomy imaging", MECO 2020
S. Corda et al., "Reduced-precision acceleration of radio-astronomical imaging on reconfigurable hardware", IEEE ACCESS 2022

Radio-astronomical imaging



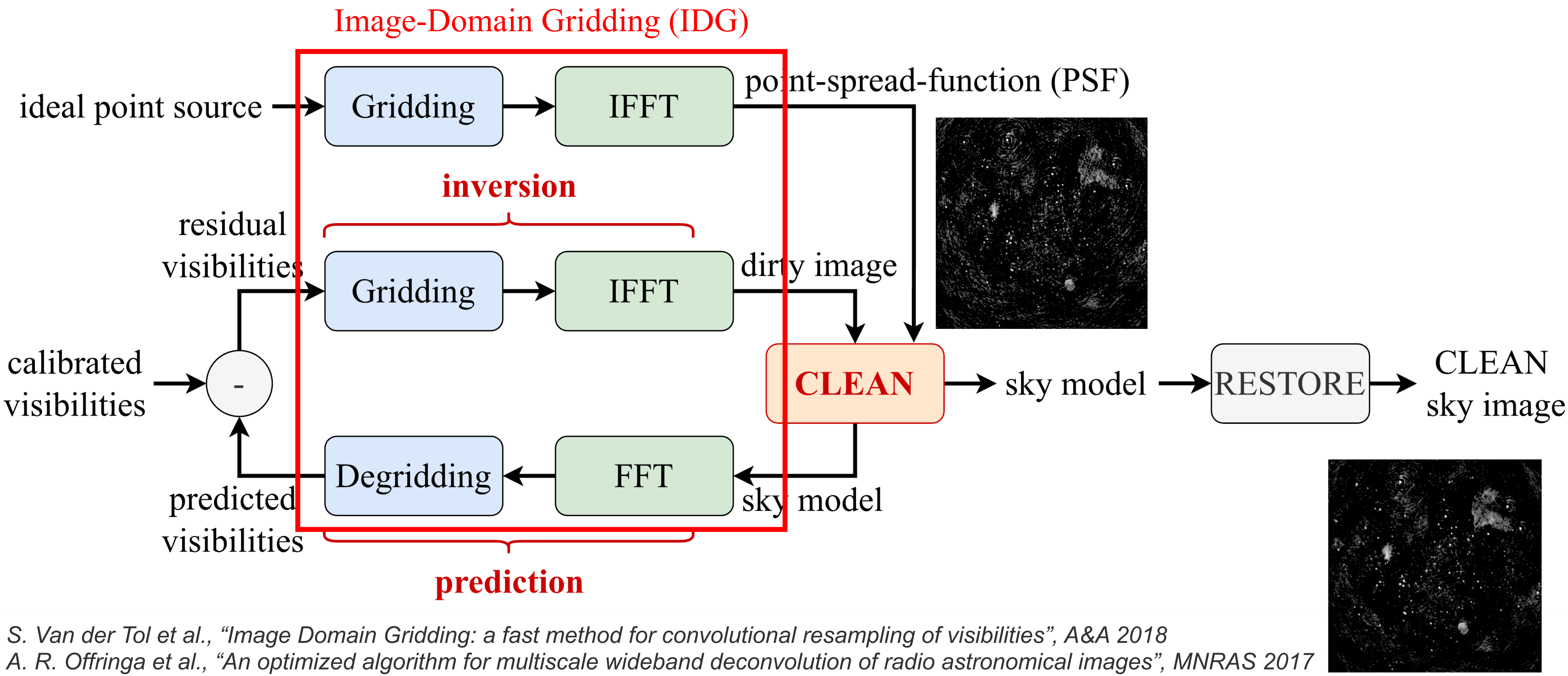
S. Van der Tol et al., "Image Domain Gridding: a fast method for convolutional resampling of visibilities", A&A 2018
A. R. Offringa et al., "An optimized algorithm for multiscale wideband deconvolution of radio astronomical images", MNRAS 2017
S. Corda et al, "Near memory acceleration on high resolution radio astronomy imaging", MECO 2020
S. Corda et al., "Reduced-precision acceleration of radio-astronomical imaging on reconfigurable hardware", IEEE ACCESS 2022

Radio-astronomical imaging



S. Van der Tol et al., "Image Domain Gridding: a fast method for convolutional resampling of visibilities", A&A 2018
A. R. Offringa et al., "An optimized algorithm for multiscale wideband deconvolution of radio astronomical images", MNRAS 2017
S. Corda et al, "Near memory acceleration on high resolution radio astronomy imaging", MECO 2020
S. Corda et al., "Reduced-precision acceleration of radio-astronomical imaging on reconfigurable hardware", IEEE ACCESS 2022

Radio-astronomical imaging



S. Van der Tol et al., "Image Domain Gridding: a fast method for convolutional resampling of visibilities", A&A 2018
A. R. Offringa et al., "An optimized algorithm for multiscale wideband deconvolution of radio astronomical images", MNRAS 2017
S. Corda et al, "Near memory acceleration on high resolution radio astronomy imaging", MECO 2020
S. Corda et al., "Reduced-precision acceleration of radio-astronomical imaging on reconfigurable hardware", IEEE ACCESS 2022

Performance Metrics

The visibilities, which are correlations of station signals, contain information on amplitude and phase of the source.

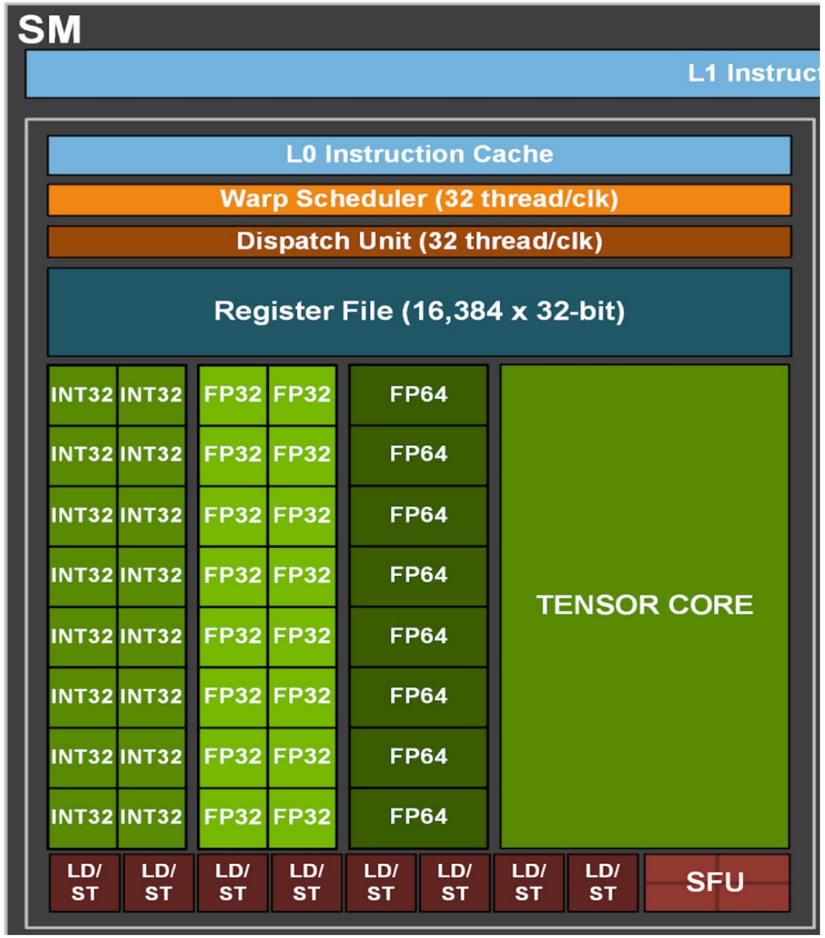
- **MVis/s**: number of Mega Visibilities per second (throughput)
- **MVis/J**: number of Mega Visibilities per Joule (energy efficiency)

Optimizations

```
Input: visibilities, wavenumbers, uvw, uvw_offset, lmn
Result: subgrids
subgrids ⟵ 0;
for s = 1...S do
    for p = 1...NxN do
        complex<float> pixel[pol] ⟵ 0;
        float lmn [3] ⟵ lmn[p]
        float phase_offset ⟵ compute_phase_offset(uvw_offsets, lmn)
        for t = 1...T do
            float phase_index ⟵ compute_phase_index(uvw, lmn)
            for c = 1...C do
                float phase ⟵ compute_phase(phase_index, phase_offset, wavenumbers)
                float phasor [2] ⟵ cosisin(phase)
                for pol in polarizations do
                    complex<float> pixel[pol] += visibilities[t][c][pol] * phasor
    apply_aterm(subgrid);
    apply_taper(subgrid);
```

Optimizations

v2

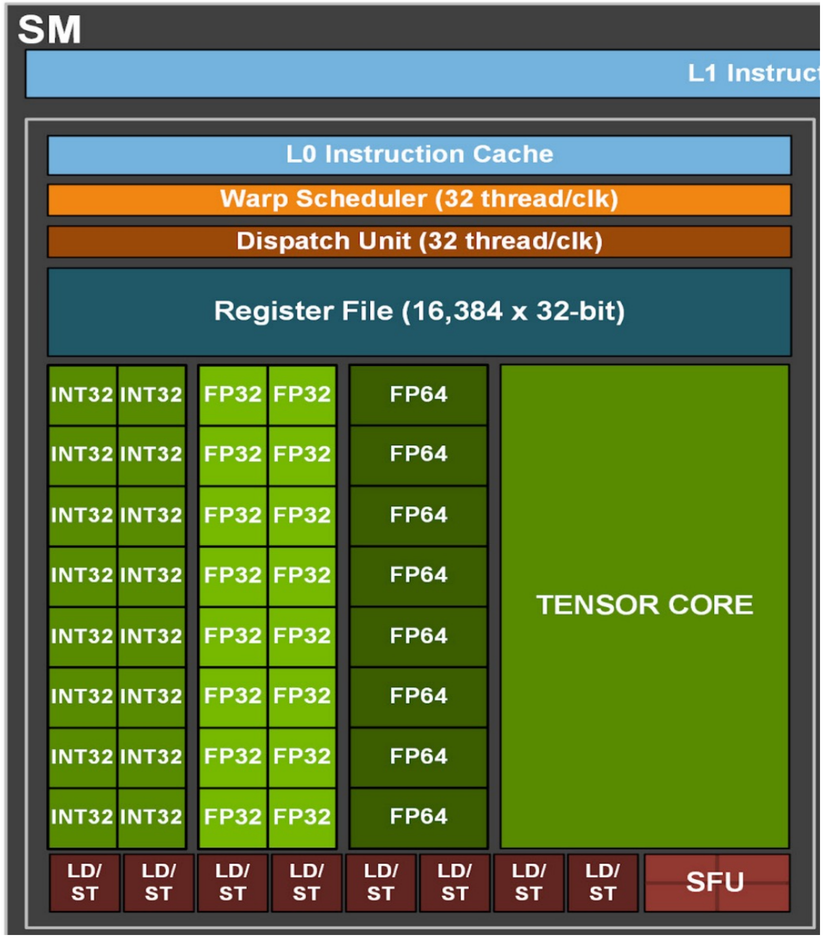


NVIDIA A100 whitepaper

```
Input: visibilities, wavenumbers, uvw, uvw_offset, lmn
Result: subgrids
subgrids ⟵ 0;
for s = 1...S do
    for p = 1...NxN do
        complex<float> pixel[pol] ⟵ 0;
        float lmn [3] ⟵ lmn[p]
        float phase_offset ⟵ compute_phase_offset(uvw_offsets, lmn)
        for t = 1...T do
            float phase_index ⟵ compute_phase_index(uvw, lmn)
            for c = 1...C do
                float phase ⟵ compute_phase(phase_index, phase_offset, wavenumbers)
                float phasor [2] ⟵ cosisin(phase)
                for pol in polarizations do
                    complex<float> pixel[pol] += visibilities[t][c][pol] * phasor
    apply_aterm(subgrid);
    apply_taper(subgrid);
```

Optimizations

v2



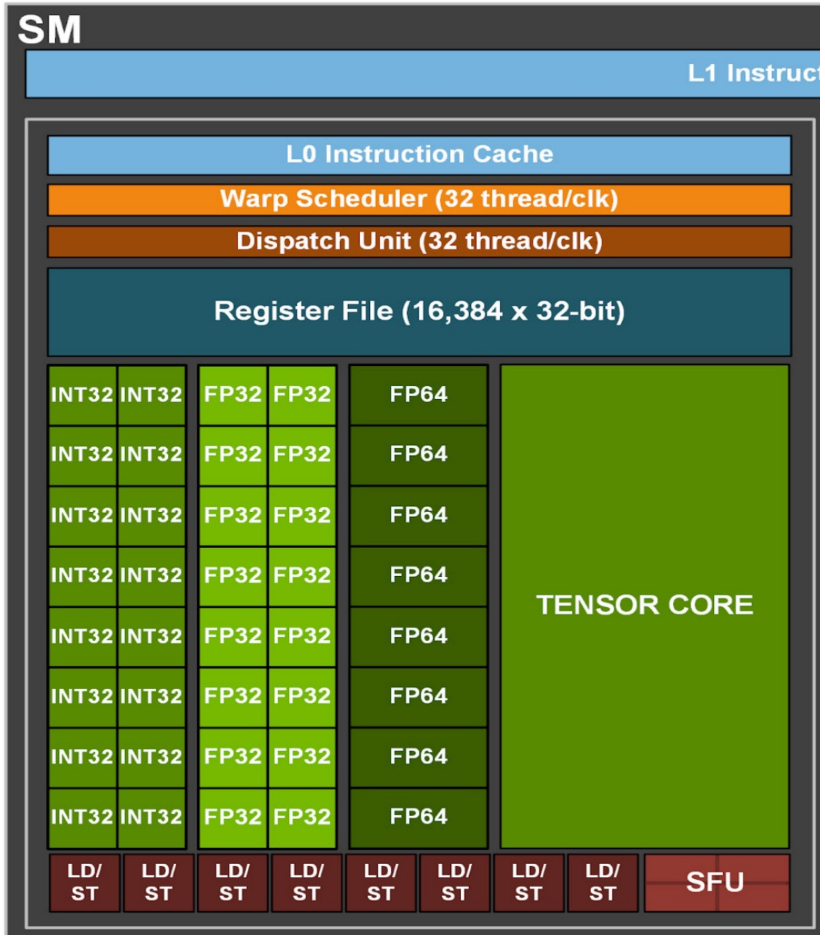
NVIDIA A100 whitepaper

v5/v6

```
Input: visibilities, wavenumbers, uvw, uvw_offset, lmn
Result: subgrids
subgrids ⟵ 0;
for s = 1...S do
    for p = 1...NxN do
        complex<float> pixel[pol] ⟵ 0;
        float lmn [3] ⟵ lmn[p]
        float phase_offset ⟵ compute_phase_offset(uvw_offsets, lmn)
        for t = 1...T do
            float phase_index ⟵ compute_phase_index(uvw, lmn)
            for c = 1...C do
                float phase ⟵ compute_phase(phase_index, phase_offset, wavenumbers)
                float phasor [2] ⟵ cosisin(phase)
                for pol in polarizations do
                    complex<float> pixel[pol] += visibilities[t][c][pol] * phasor
    apply_aterm(subgrid);
    apply_taper(subgrid);
```

Optimizations

v2



NVIDIA A100 whitepaper

v5/v6

Input: visibilities, wavenumbers, uvw, uvw_offset, lmn
Result: subgrids

```
subgrids ⟵ 0;
for s = 1...S do
    for p = 1...NxN do
        complex<float> pixel[pol] ⟵ 0;
        float lmn [3] ⟵ lmn[p]
        float phase_offset ⟵ compute_phase_offset(uvw_offsets, lmn)
        for t = 1...T do
            float phase_index ⟵ compute_phase_index(uvw, lmn)
            for c = 1...C do
                float phase ⟵ compute_phase(phase_index, phase_offset, wavenumbers)
                float phasor [2] ⟵ cosisin(phase)
                for pol in polarizations do
                    complex<float> pixel[pol] += visibilities[t][c][pol] * phasor
    apply_aterm(subgrid);
    apply_taper(subgrid);
```

v9

Original

```
float vxi = 0.0f, vyi = 0.0f, vzi = 0.0f;
  for (int j = hipThreadIdx_x; j < count1; j += hipBlockDim_x) {
    float dx = xx1[j] - xxi;
    float dy = yy1[j] - yyi;
    float dz = zz1[j] - zzi;
    float dist2 = dx*dx + dy*dy + dz*dz;
    if (dist2 < fsrrmax2) {
      float rtemp = (dist2 + rsm2)*(dist2 + rsm2)*(dist2 + rsm2);
      float f_over_r = massi*mass1[j]*(1.0f/sqrt(rtemp) - (ma0 +
dist2*(ma1 + dist2*(ma2 + dist2*(ma3 + dist2*(ma4 + dist2*ma5))))));

      vxi += fcoeff*f_over_r*dx;
      vyi += fcoeff*f_over_r*dy;
      vzi += fcoeff*f_over_r*dz;
    }
  }
```

Modified to use Packed FMA32

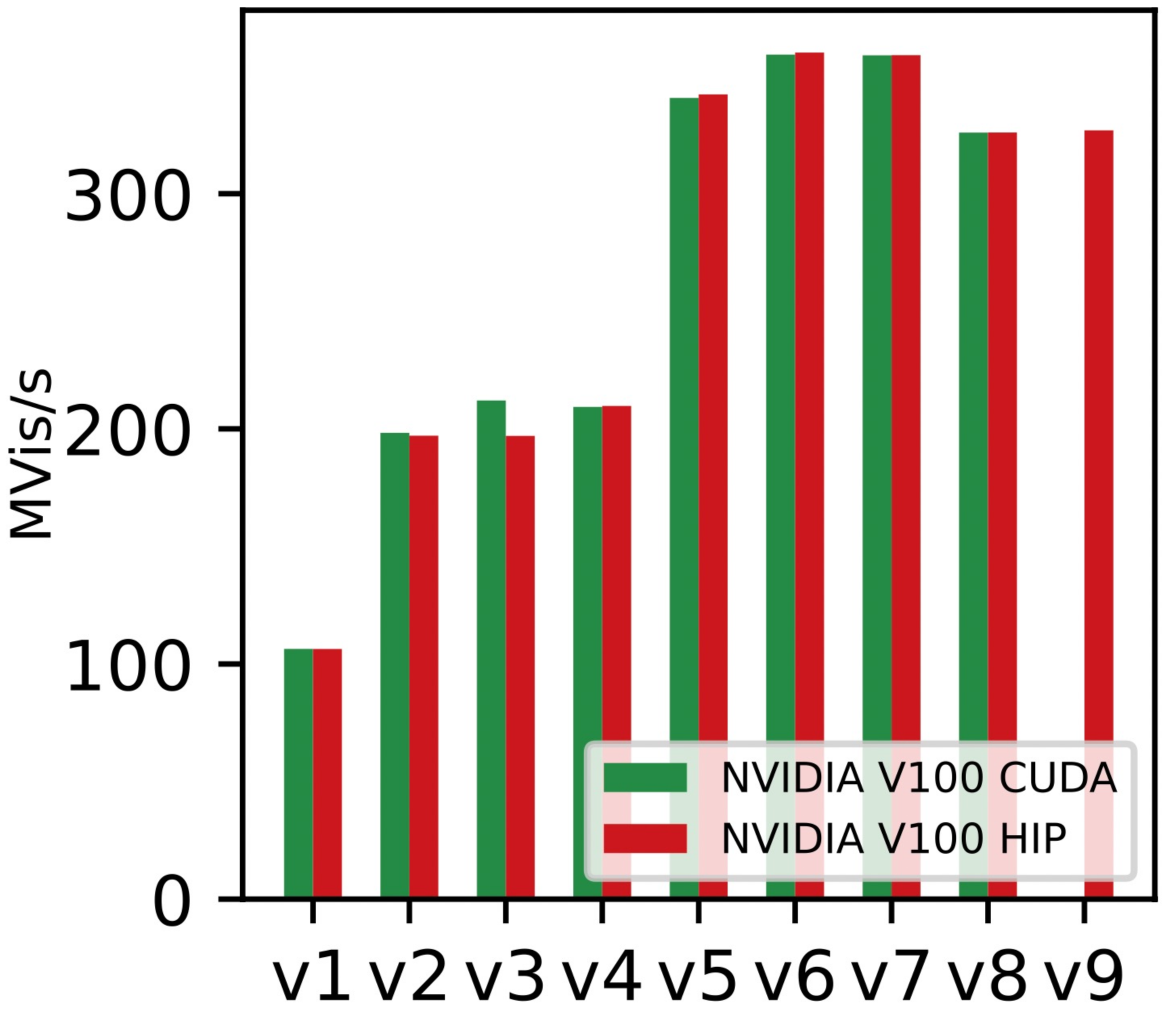
```
float2 vxi = 0.0f, vyi = 0.0f, vzi = 0.0f;
  for (int j = hipThreadIdx_x; j < count1; j += 2*hipBlockDim_x) {
    float2 dx = {xx1[j] - xxi, xx1[j+ hipBlockDim_x] - xxi};
    float2 dy = {yy1[j] - yyi, yy1[j+ hipBlockDim_x] - yyi};
    float2 dz = {zz1[j] - zzi, zz1[j+ hipBlockDim_x] - zzi};
    float2 dist2 = dx*dx + dy*dy + dz*dz;
    if (dist2 < fsrrmax2) {
      float2 rtemp = (dist2 + rsm2)*(dist2 + rsm2)*(dist2 + rsm2);
      float2 f_over_r = massi*mass1[j]*(1.0f/sqrt(rtemp) - (ma0 +
dist2*(ma1 + dist2*(ma2 + dist2*(ma3 + dist2*(ma4 + dist2*ma5))))));

      vxi += fcoeff*f_over_r*dx;
      vyi += fcoeff*f_over_r*dy;
      vzi += fcoeff*f_over_r*dz;
    }
  }
```

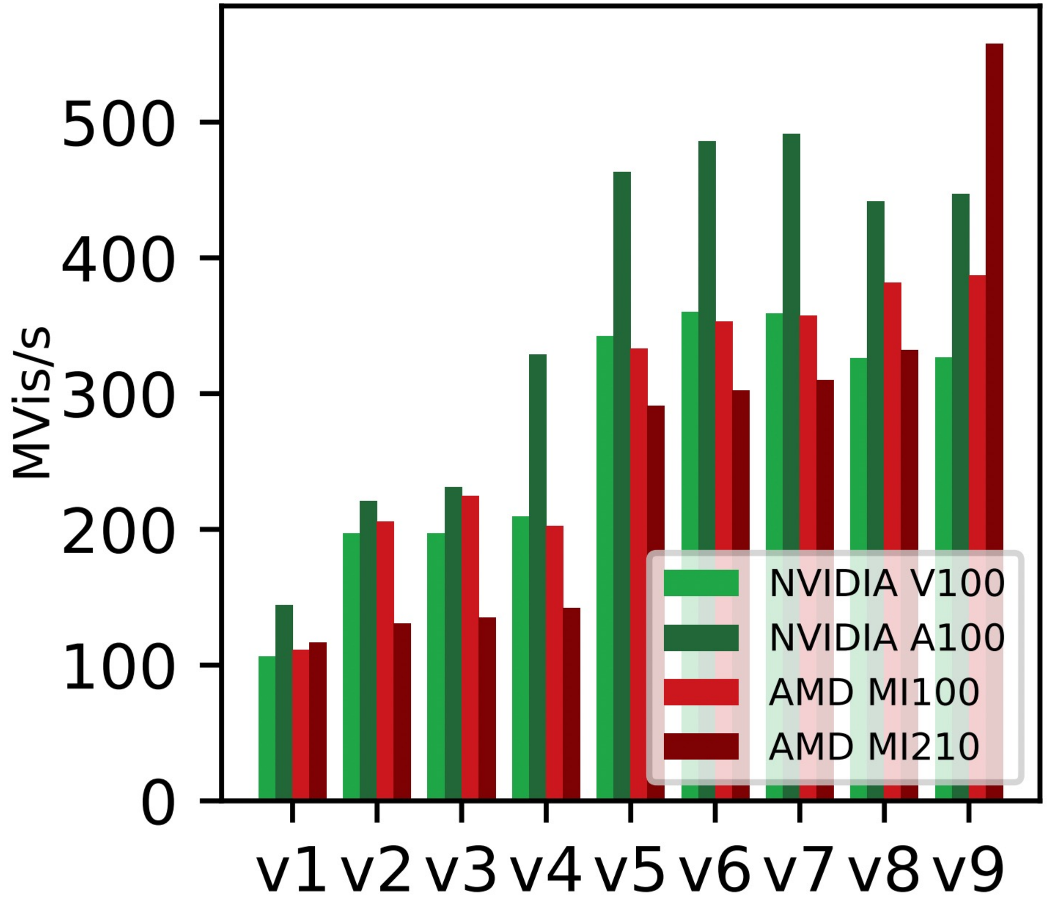
AMD CNDA2 whitepaper

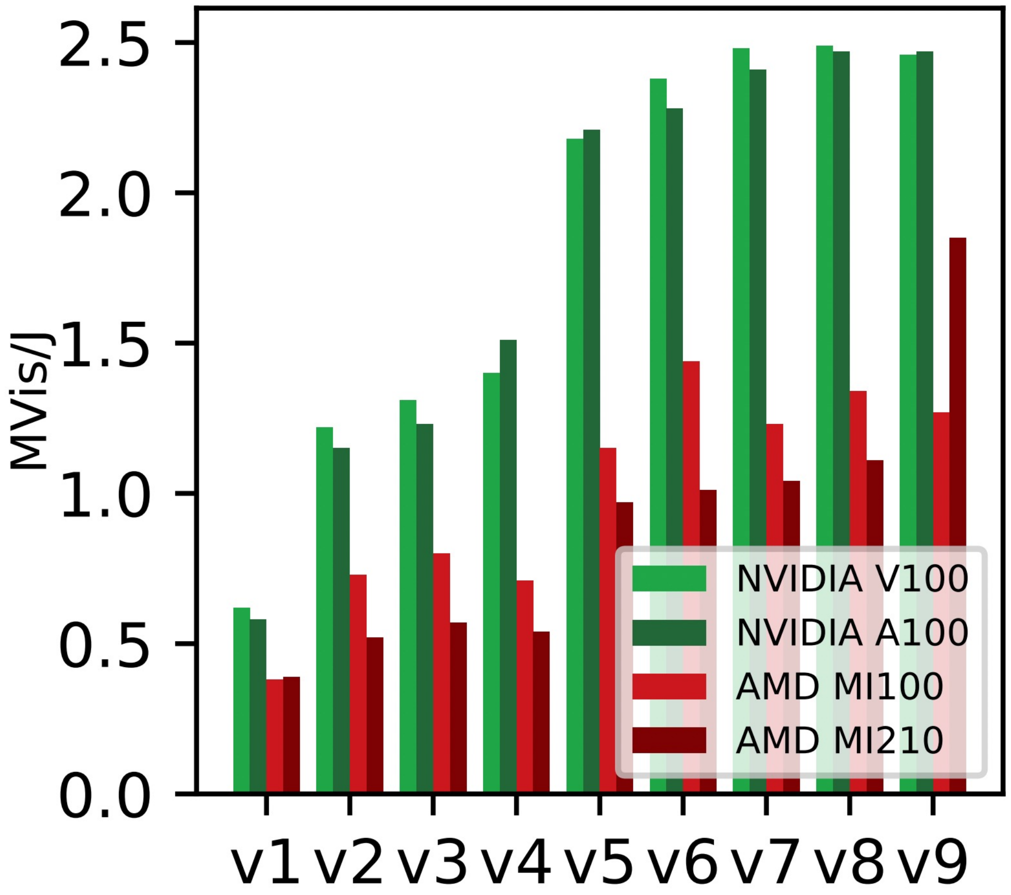
V100: CUDA vs HIP

- Marginal differences
- Certain optimization are better implemented by HIP and others by NVIDIA compiler



Performance (gridder)

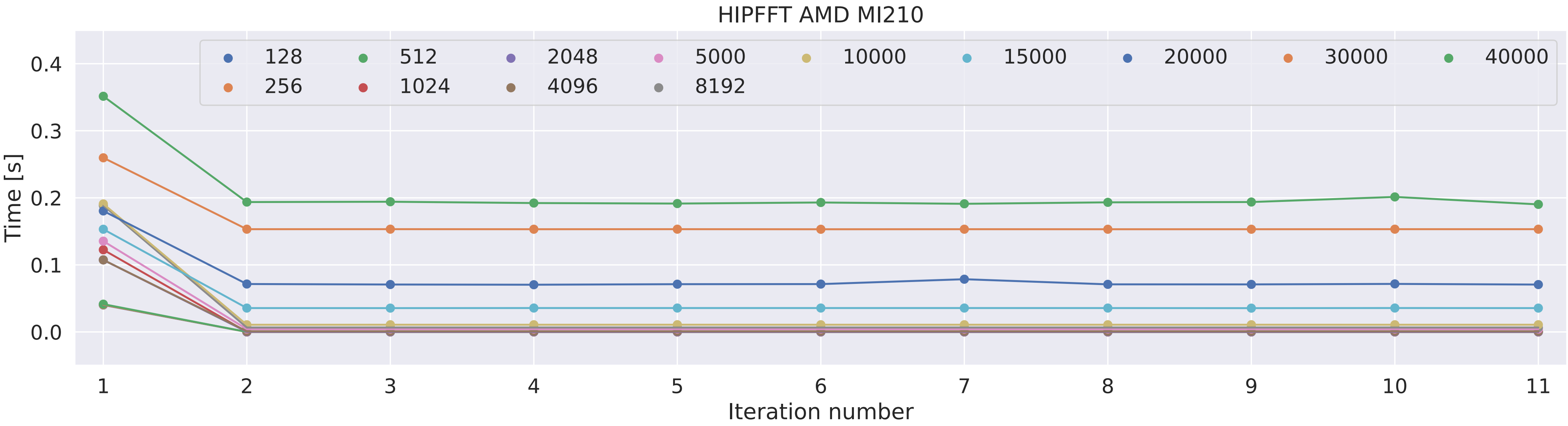




- CDNA2 shows good potential by exploiting FP32 packed instructions (v9)
- Application code needs additional tuning to achieve better performance

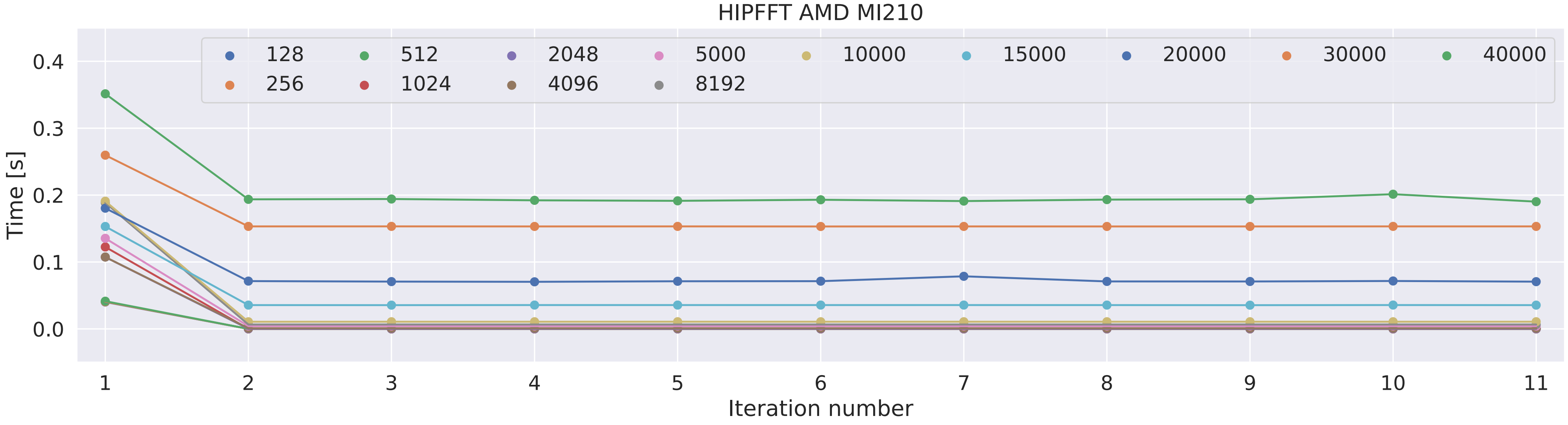
S. Corda, B. Veenboer and E. Tolley, "PMT: Power Measurement Toolkit", accepted at Supercomputing Conference 2022 (HUST workshop)

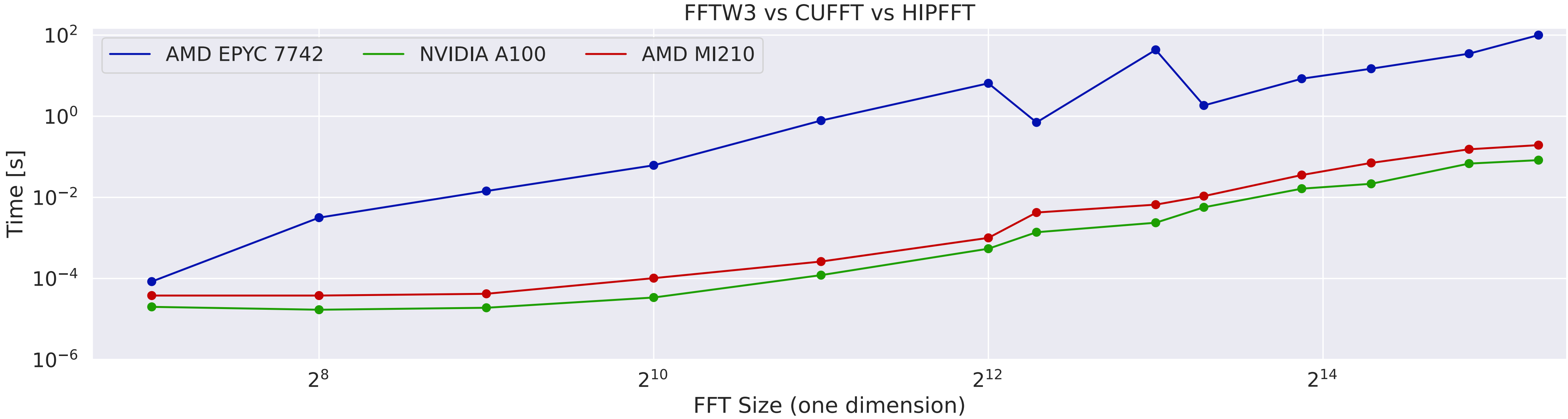
Large 2D FFTs



- "Cold run" issue

Large 2D FFTs





- "Cold run" issue
- GPUs outperform CPUs

Conclusions

- HIP provides equivalent performance for the IDG gridder/degridder on NVIDIA GPUs

- HIP provides "acceptable" performance portability to AMD GPUs
  1. Easier than maintaining both a CUDA and OpenCL codebase
  2. Additional tuning is needed to get closer to peak performance

- Recent AMD GPUs can achieve competitive performance if the application can be re-shaped to support single-precision packed instructions

- We step on several issues running HIPFFT (performance on AMD and compatibility with NVIDIA).

Conclusions

- HIP provides equivalent performance for the IDG gridder/degridder on NVIDIA GPUs

- HIP provides "acceptable" performance portability to AMD GPUs
  1. Easier than maintaining both a CUDA and OpenCL codebase
  2. Additional tuning is needed to get closer to peak performance

- Recent AMD GPUs can achieve competitive performance if the application can be re-shaped to support single-precision packed instructions

- We step on several issues running HIPFFT (performance on AMD and compatibility with NVIDIA).

- Next Steps:
  - Gridder/Degridder performance difference analysis and optimization
  - Discuss with AMD HIPFFT issues
  - Port CUDA production code with HIP
  - Co-design evaluation on several AMD/NVIDIA GPU generations

Radio-astronomical imaging on GPUs and FPGAs

Thanks!

Stefano Corda

stefano.corda@epfl.ch

04-10-2022

Swiss SKA Days 2022, Lugano