Finanziato dall'Unione europea NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

GREEN HPC toward SKA era with RICK

GIOVANNI LACOPO

UNIVERSITY OF TRIESTE, OATS INAF, PSC

Supervisors: Luca Tornatore, Giuliano Taffoni, Claudio Gheller

Pawsey advisors: Ugo Varetto, Pascal Elahi, Maciej Cytowski

Green computing toward SKA era with RICK

Giovanni Lacopo[1,2][0000−1111−2222−3333], Claudio Gheller[3,4][1111−2222−3333−4444], Emanuele De Rubeis[4,3][0000−0002−0428−2055], Pascal Jahan Elahi[5][0000−0002−6154−7224], Maciej Cytowski[5], Luca Tornatore[2], Giuliano Taffoni[2], and Ugo Varetto[5]

[1] Università degli studi di Trieste, via Alfonso Valerio 2, 34127 Trieste, Italy
[2] Astronomical Observatory of Trieste INAF, via GB Tiepolo 11, 34143 Trieste, Italy
`info.oats@inaf.it`
[3] Istituto di Radio Astronomia, INAF, Via P. Gobetti 101, 40129 Bologna, Italy
[4] Dipartimento di Fisica e Astronomia, Università di Bologna, Via P. Gobetti 92/3, 40129 Bologna, Italy
[5] Pawsey Supercomputing Centre, 1 Bryce Avenue, Kensington WA 6151, Australia
`admin@pawsey.org.au`

Abstract. Square Kilometer Array is expected to generate hundreds of petabytes of data per year, two orders of magnitude more than current radio interferometers. Data processing at this scale necessitates advanced High Performance Computing (HPC) resources. However, modern HPC platforms consume up to tens of MW, i.e. megawatts, and energy-to-solution in algorithms will become of utmost importance in the next future. In this work we study the trade-off between energy-to-solution and time-to-solution of our **RICK** code (Radio Imaging Code Kernels), which is a novel approach to implement the w-stacking algorithm designed to run on state-of-the-art HPC systems. The code can run on heterogeneous systems exploiting the accelerators. We did both single-node tests and multi-node tests with both CPU and GPU solutions, in order to study which one is the greenest and which one is the fastest. We then defined the **green productivity**, i.e. a quantity which relates energy-to-solution and time-to-solution in different code configurations compared to a reference one. Configurations with the highest green productivities are the most efficient ones. The tests have been run on the Setonix machine available at the Pawsey Supercomputing Research Centre (PSC) in Perth (WA), ranked as 28^{th} in Top500[6] list, updated at June 2024.

Keywords: Green computing · Radio astronomy · Data analysis.

SKA-Low challenges

- ~ 1 TB/s data delivered
- ~ 300 PB of data per year

How to deal with such amount of data?

Possible solutions

- Single node OpenMP/GPU? → I/O dominated
- Pure MPI? → Communication dominated

In both cases the code is embarrassingly memory bound!!

Best solutions

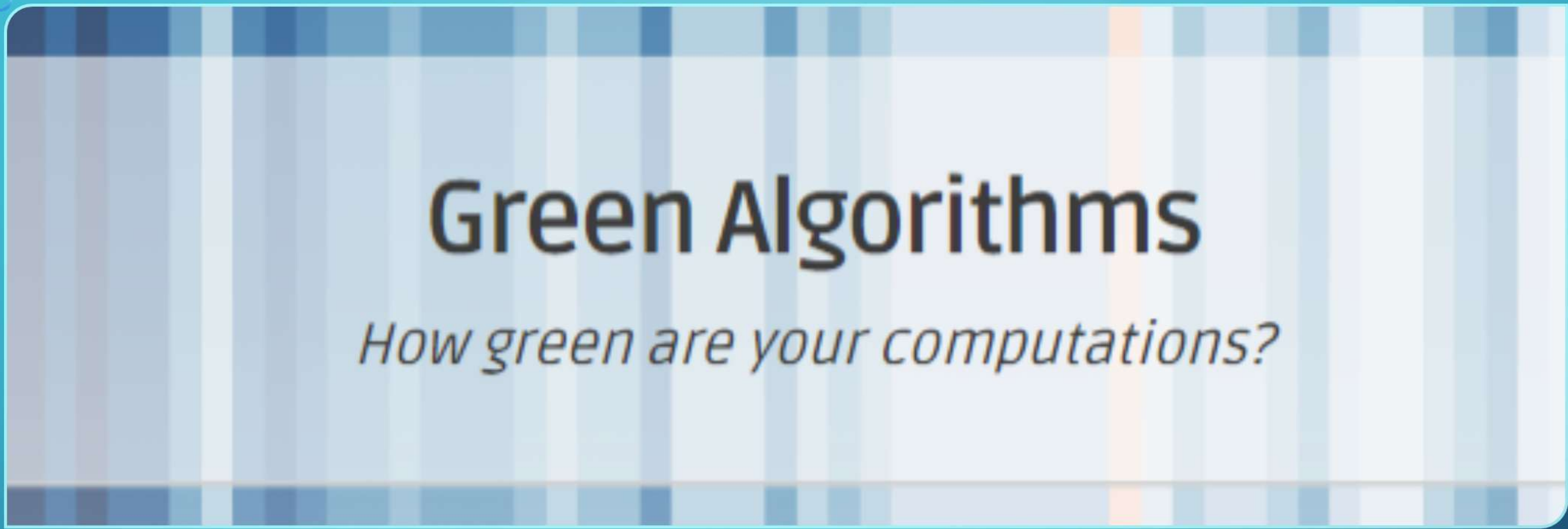
- Hybrid MPI/OpenMP → The communication surface is reduced!
- MPI/GPU → Exploit the GPUs computing power without caring about memory requirements!

THE NEED FOR GREEN COMPUTING

Pure performance is not the only thing we must consider!!!

CURRENT EXASCALE PLATFORMS CONSUME TENS OF MEGAWATTS UNDER FULL WORKLOAD!!!

SOLUTION 1) NUCLEAR POWER PLANTS



SOLUTION 2) GREEN ALGORITHMS & GREEN COMPUTING

RADIOASTRONOMY SOFTWARE

Working in-between radioastronomy HPC, I am focusing on:

- trying to enable the software used for the processing and analysis of radio data to effectively exploit supercomputing solutions,
- address the challenge posed by increasingly larger and complex datasets.
- Providing solutions to improve current codes (NOT to replace them)
- Best trade-off between the computing resources used and the performance

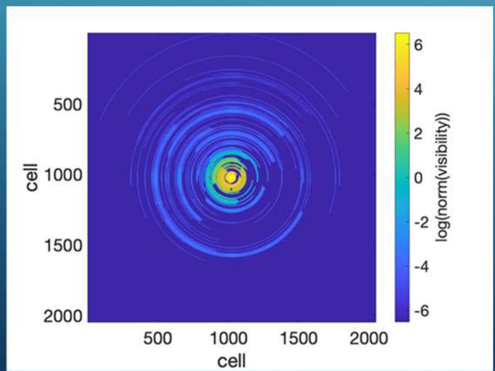
CASE STUDY: IMAGING

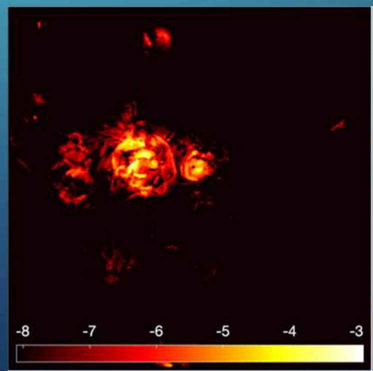
Essentially, we want to invert the following integral:

$$V(u,v,w) = \int\int \frac{I(l,m)}{\sqrt{1-l^2-m^2}} \times e^{-2\pi i\left(ul+vm+w\left(\sqrt{1-l^2-m^2}-1\right)\right)} dl\,dm$$

that **maps** the visibilities V measured from the interferometer to the sky brightness I, providing the actual image of the sky.

$(u,\ v,\ w)$ are the baselines coordinates, and $(l,\ m)$ are the sky coordinates.



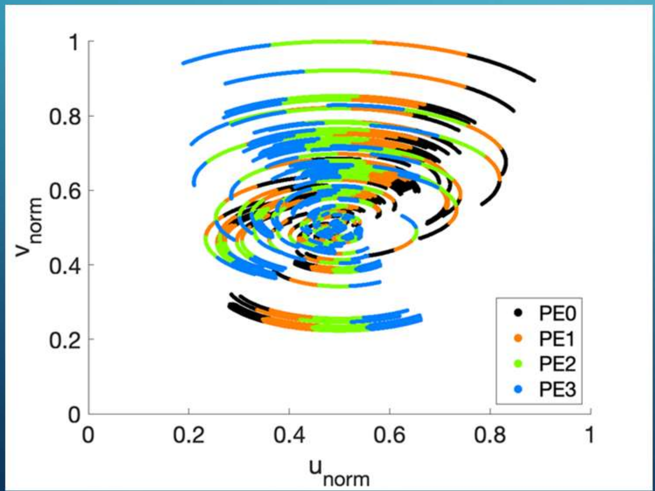


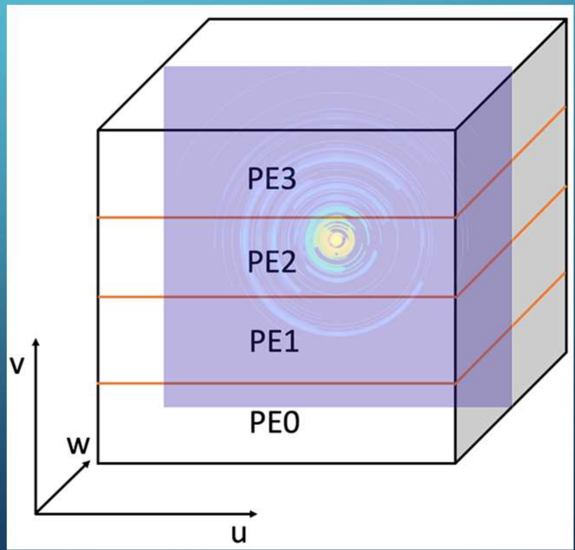
OVERCOMING THE MEMORY WALL

Parallel computing allows to use multiple processors distributing data among their memory:

- Visibilities (and work) are evenly distributed among processing units
- The mesh is split among processing units. The full mesh is never stored in a single memory

 Problems of "any" size can be supported





Main issue:

visibilities are distributed across memories unrelated to mesh slabs

 Lots of communication required

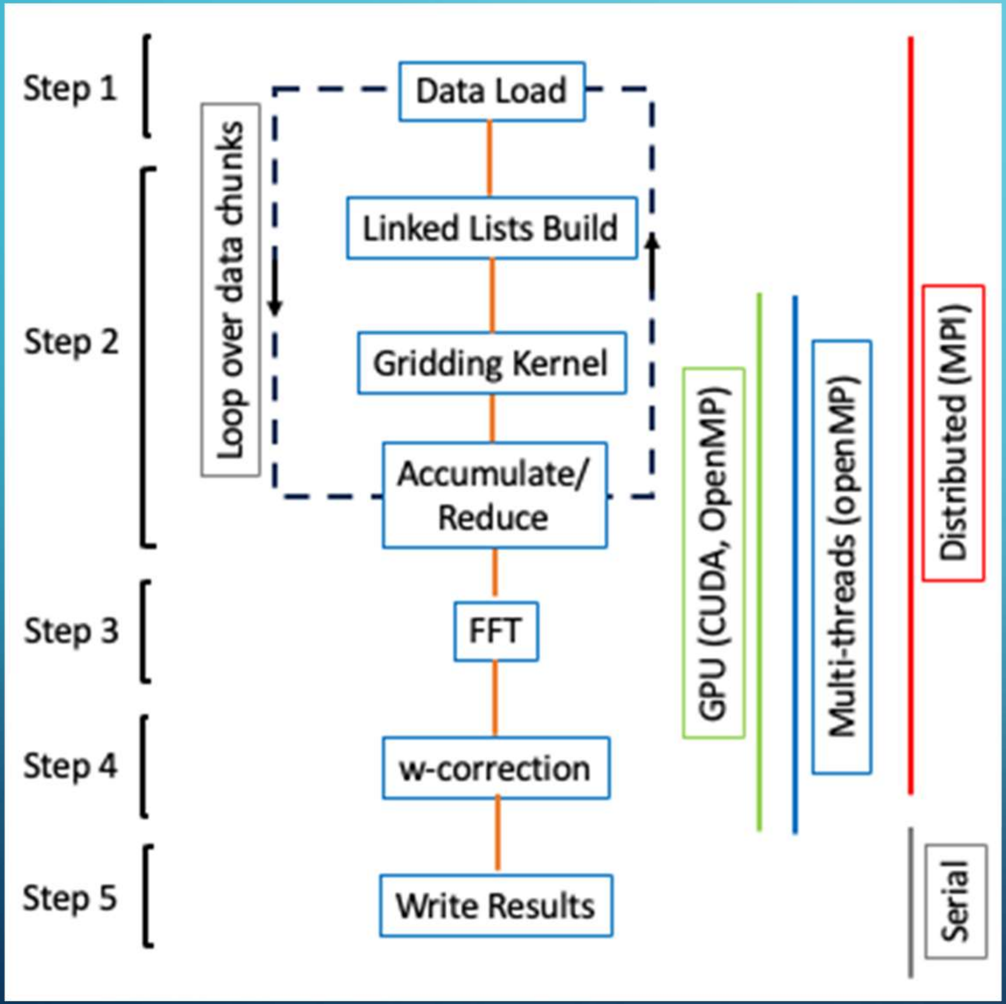
IMAGING: MAIN STEPS

Imaging requires essentially 2 (+1) **computational demanding** operations:

1. **Discretization** of the problem → map visibilities on a regular mesh (needed for FFT) + weighting + tapering
2. **FFT** transform from Fourier to Real space
3. (**W–correction**, if needed, to correct for Earth curvature)

These operations represent computational demanding problems that can benefit from HPC.

RICK HPC support



The code is publicly available at
https://www.ict.inaf.it/gitlab/claudio.gheller/hpc_imaging

TEST CASE

LOFAR HBA Inner Station:

- 1891 baselines
- 121-169 MHz, 4 polarizations
- 8 hrs

Data:

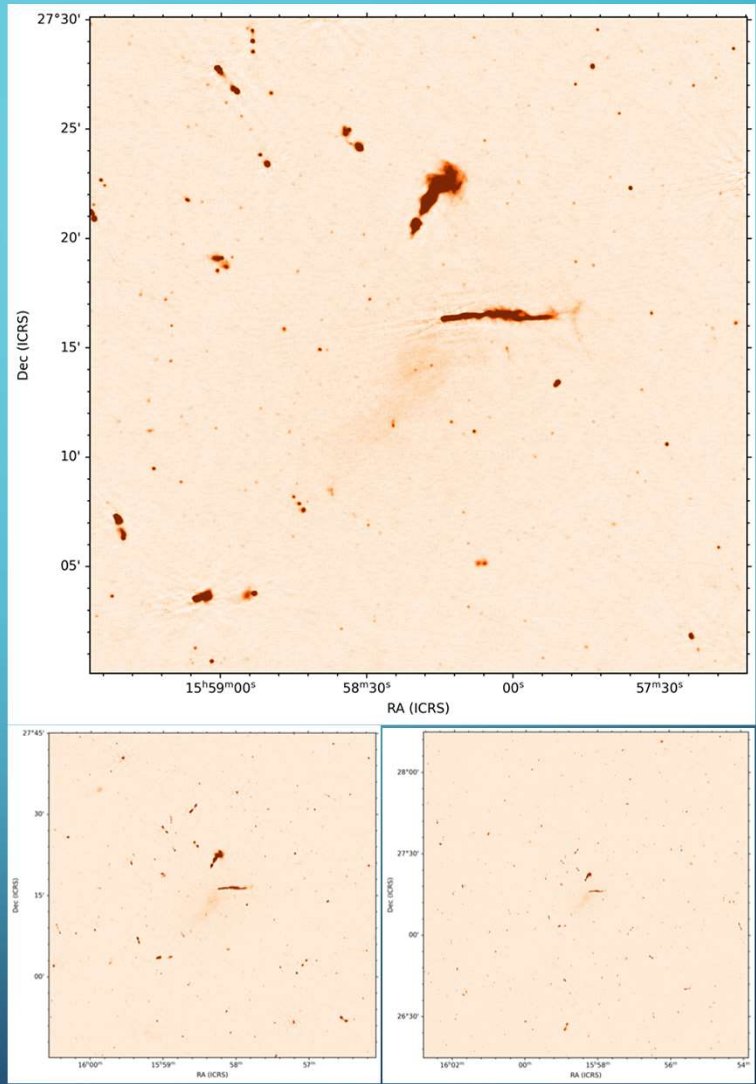
- 543 million visibilities
- 4.5 GB

Mesh/Image:

- 4096x4096 px, 80 MB
- 4096x4096x16, 430 MB

Memory Usage

- ~ 10 GB



Curtesy of Luca Bruno

TEST CASE

TEST 1 (MULTI-NODE)

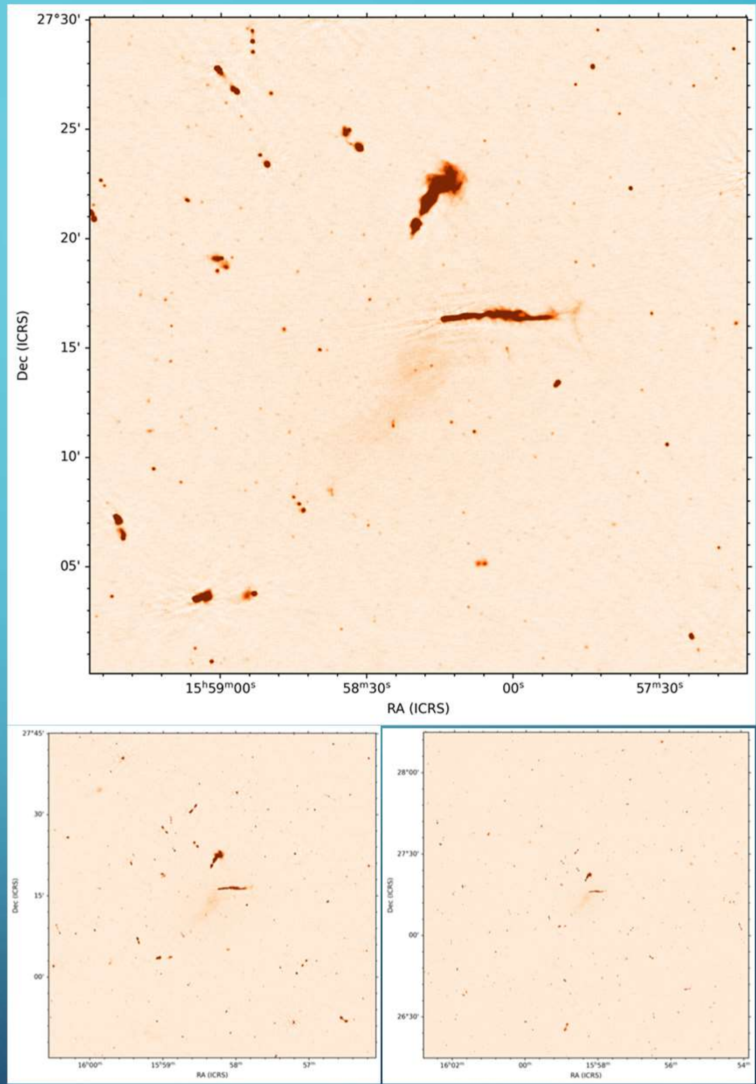
Stacking of 18 frequencies obs. (~80 GB)

Grid size 16384x16384x24

TEST 2 (SINGLE-NODE)

Stacking of 2 frequencies obs. (~9 GB)

Grid size 4096x4096x64

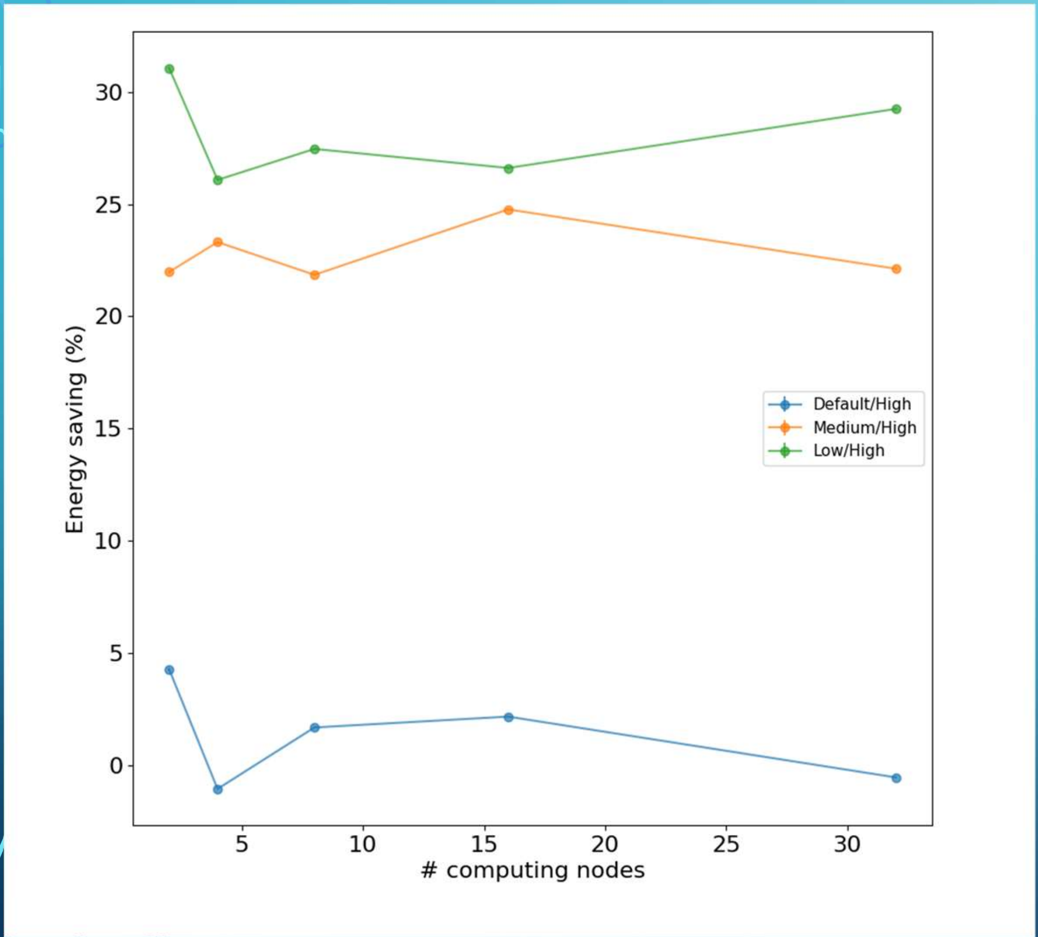


Curtesy of Luca Bruno

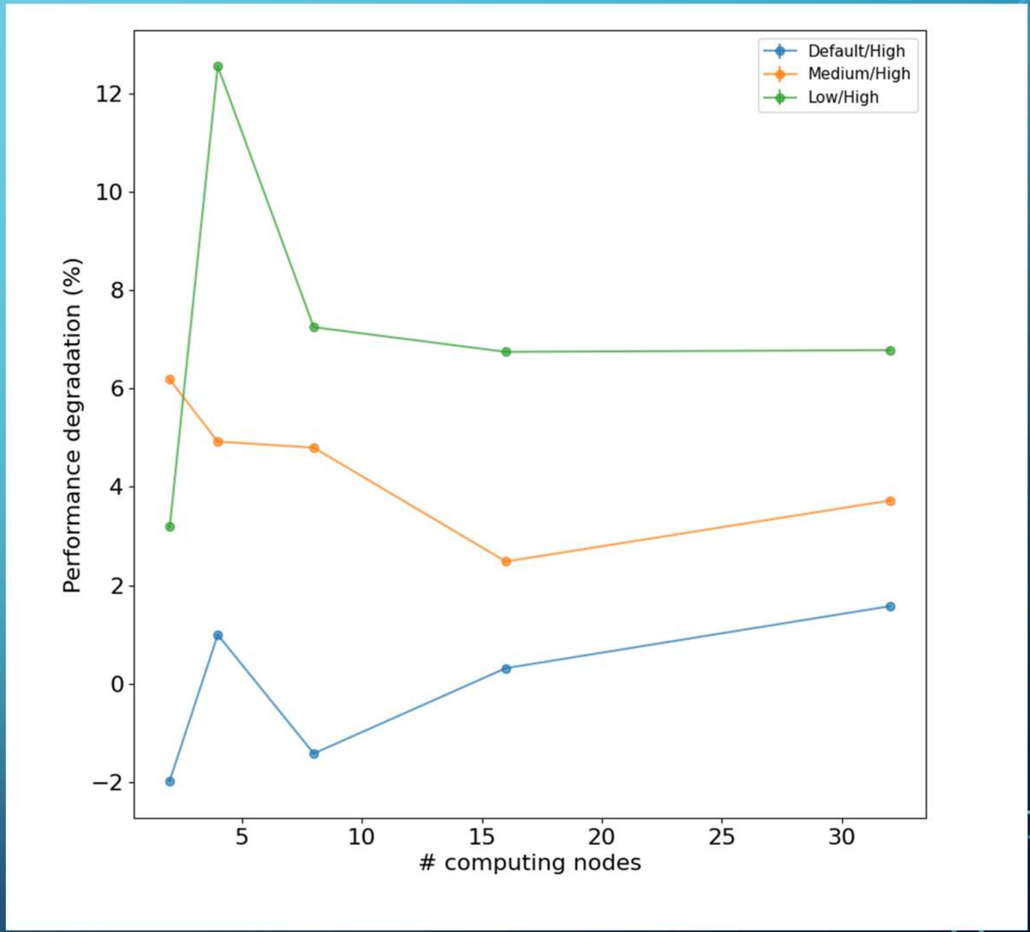
SETONIX IS THE QUOKKA'S SCIENTIFIC NAME

HELLO!!!

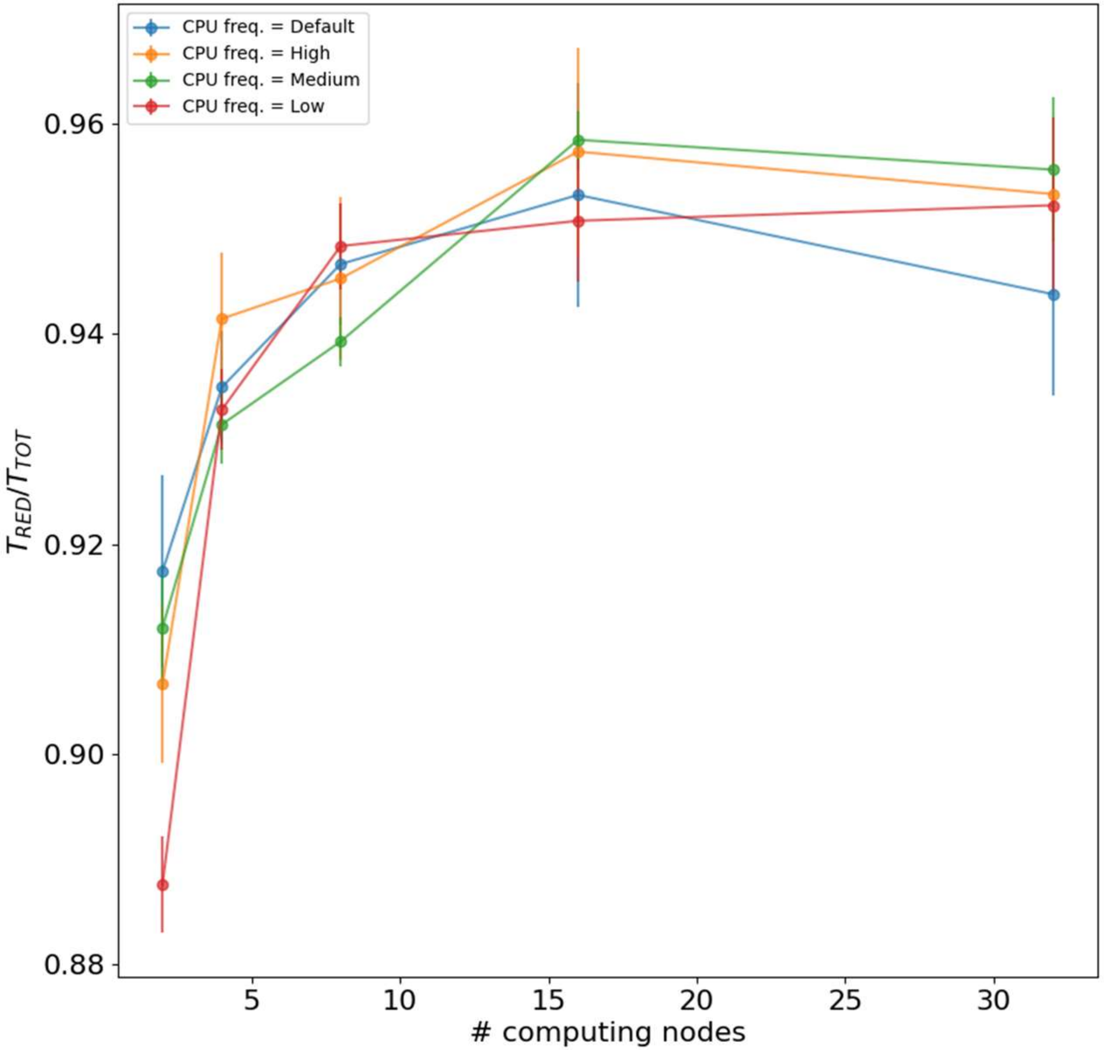
CONFIG.	TASKS	THREADS	GPUS	NODES
MPI	256	1	0	2
MPI	512	1	0	4
MPI	1024	1	0	8
MPI	2048	1	0	16
MPI	4096	1	0	32



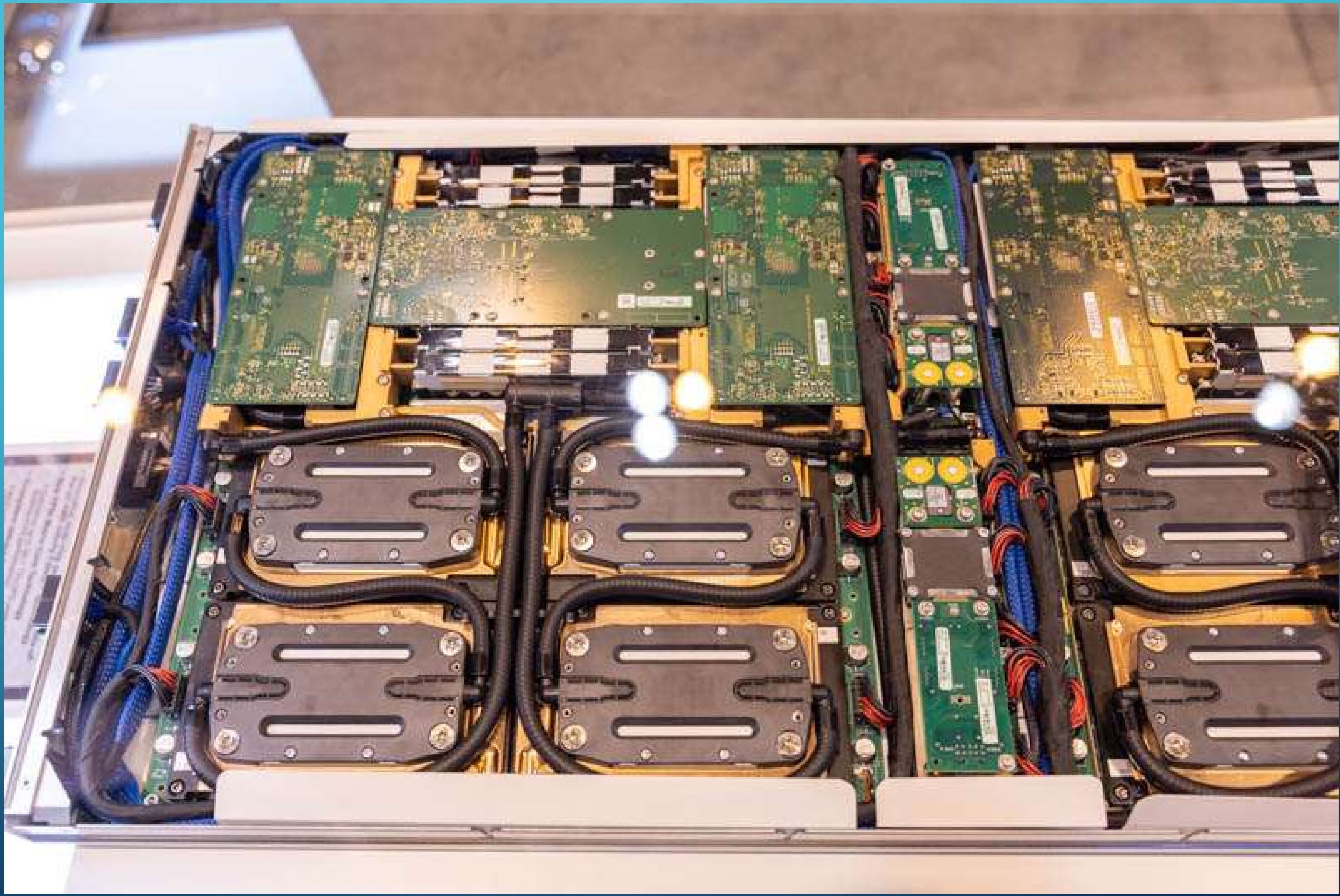
Energy saving (%)
computing nodes
Default/High
Medium/High
Low/High



Performance degradation (%)
computing nodes
Default/High
Medium/High
Low/High

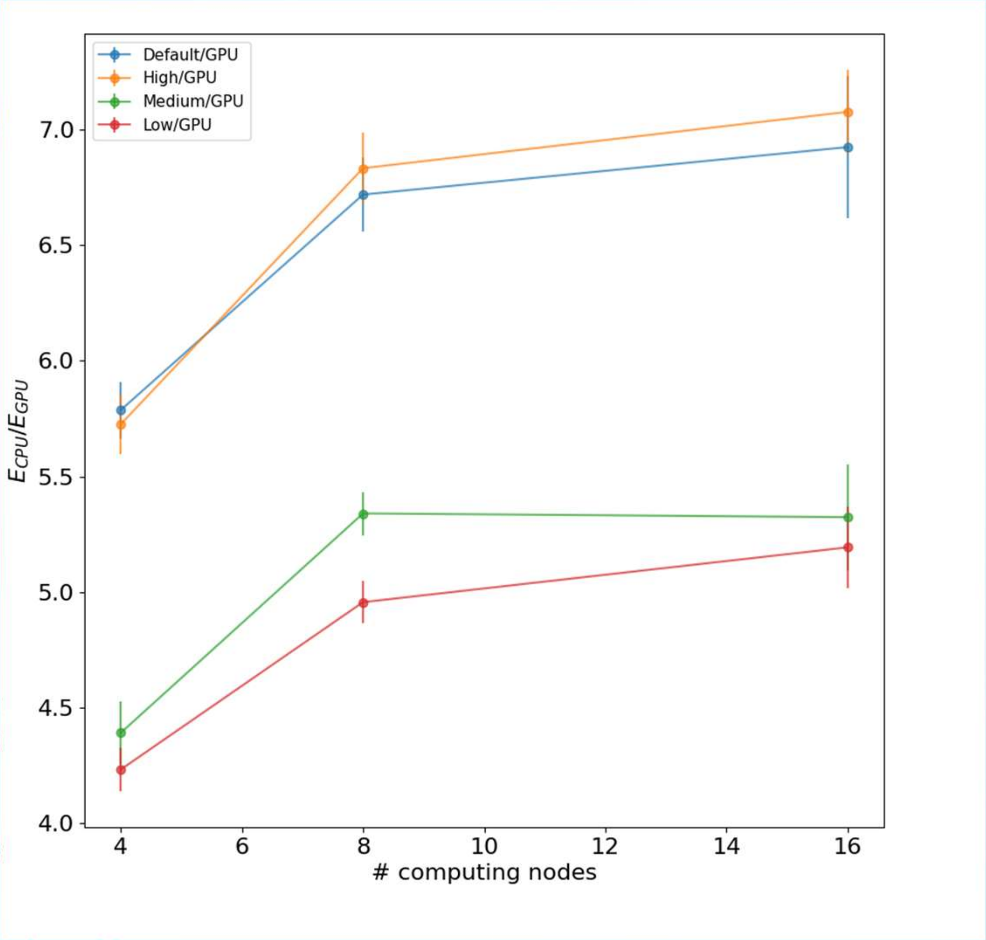

CPU freq. = Default
CPU freq. = High
CPU freq. = Medium
CPU freq. = Low
T_{RED}/T_{TOT}
0.96
0.94
0.92
0.90
0.88
5
10
15
20
25
30
computing nodes

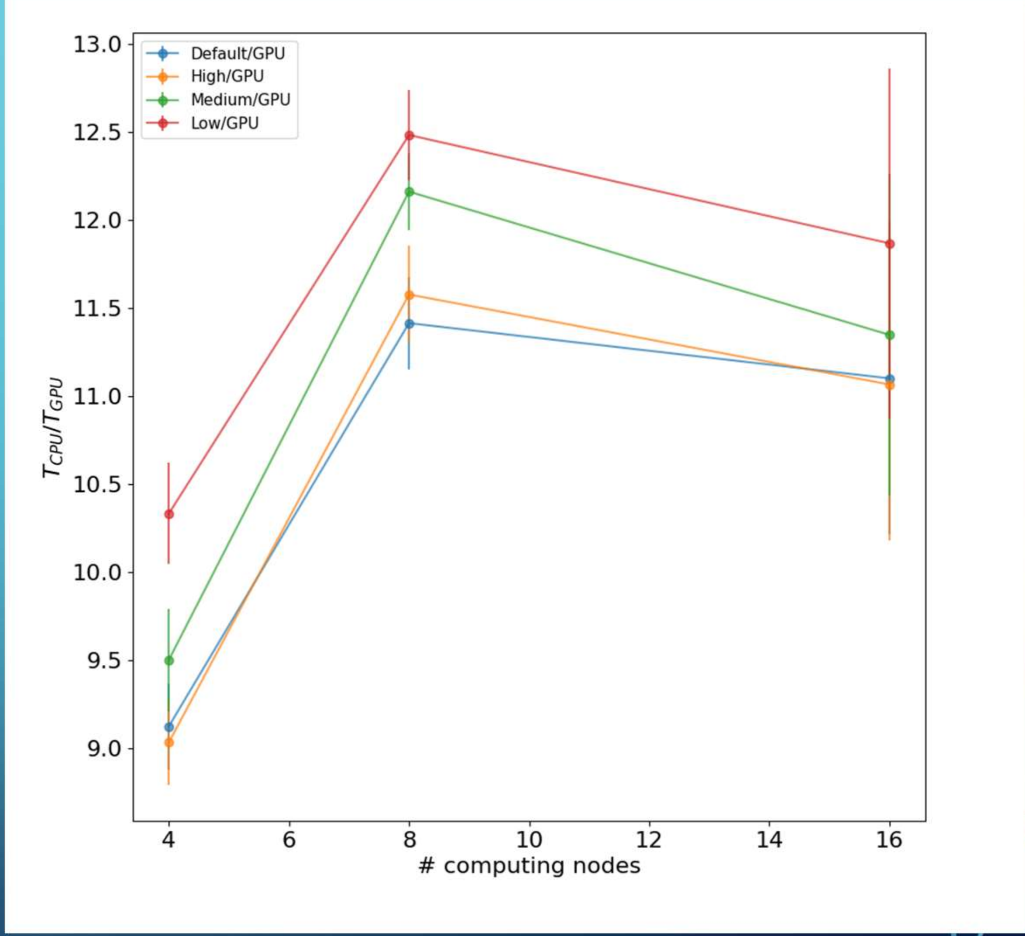
WHAT HAPPENS WITH GPUs?



CONFIG.	TASKS	THREADS	GPUS	NODES
MPI	256	1	0	2
MPI	512	1	0	4
MPI	1024	1	0	8
MPI	2048	1	0	16
MPI	4096	1	0	32

CONFIG	TASKS	THREADS	GPUS	NODES
GPU	32	1	32	4
GPU	64	1	64	8
GPU	128	1	128	16

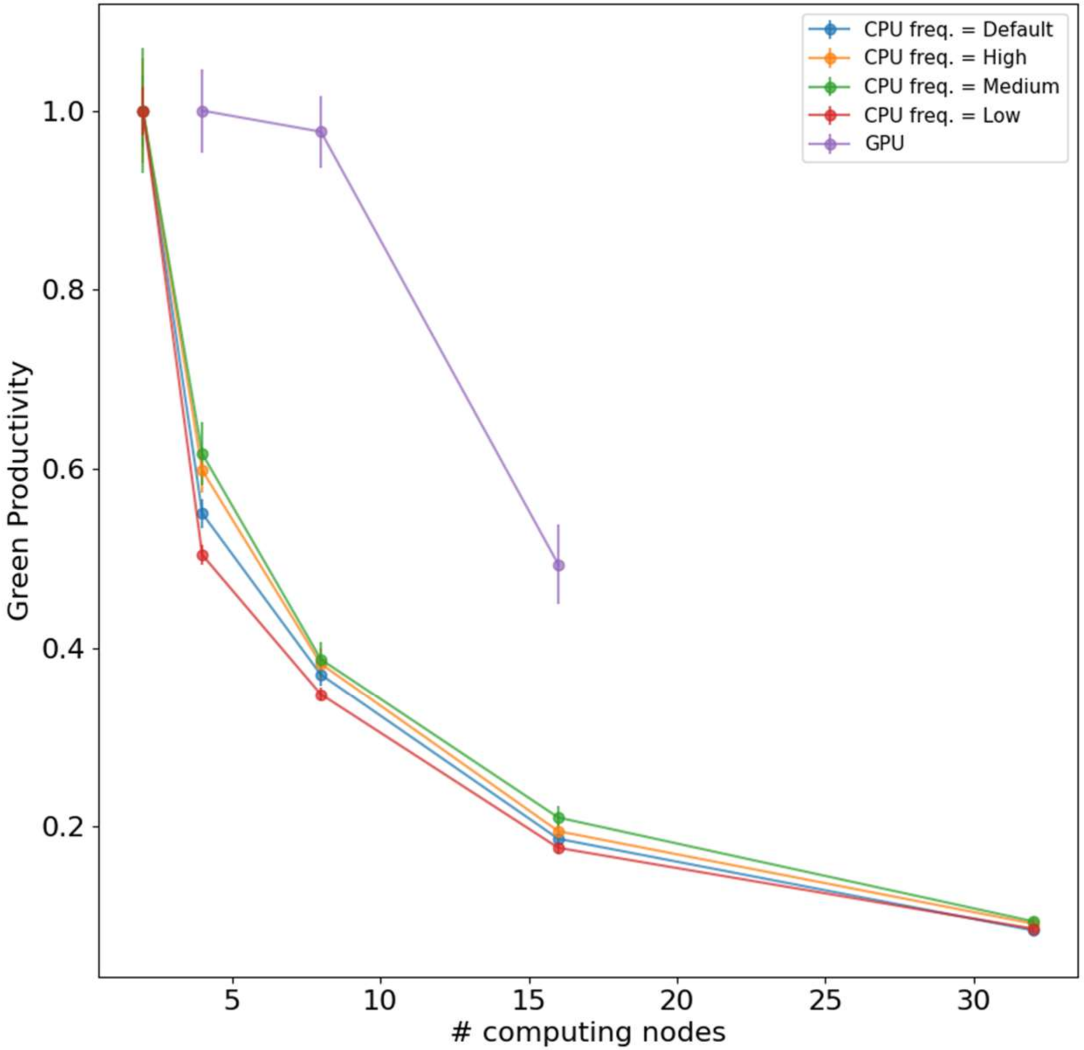

Default/GPU
High/GPU
Medium/GPU
Low/GPU
E_{CPU}/E_{GPU}
7.0
6.5
6.0
5.5
5.0
4.5
4.0
4
6
8
10
12
14
16
computing nodes


Default/GPU
High/GPU
Medium/GPU
Low/GPU
T_{CPU}/T_{GPU}
13.0
12.5
12.0
11.5
11.0
10.5
10.0
9.5
9.0
4
6
8
10
12
14
16
computing nodes

WEIGHTED GREEN PRODUCTIVITY

$$GreenProductivity = \frac{\frac{T_0}{T_N}}{w\frac{E_N}{E_0}}$$

IN THIS CASE WE CHOOSE w=1 AND WE SEARCH FOR MAXIMA IN THIS FUNCTION



CPU freq. = Default
CPU freq. = High
CPU freq. = Medium
CPU freq. = Low
GPU
Green Productivity
computing nodes
1.0
0.8
0.6
0.4
0.2
5
10
15
20
25
30

The code is definitely MEMORY-BOUND!

The best configuration in terms of Green Productivity is the one which minimizes the computing resources needed to fit the problem!!!

CONFIG	TASKS	THREADS	GPUS	NODES
MPI	128	1	0	1

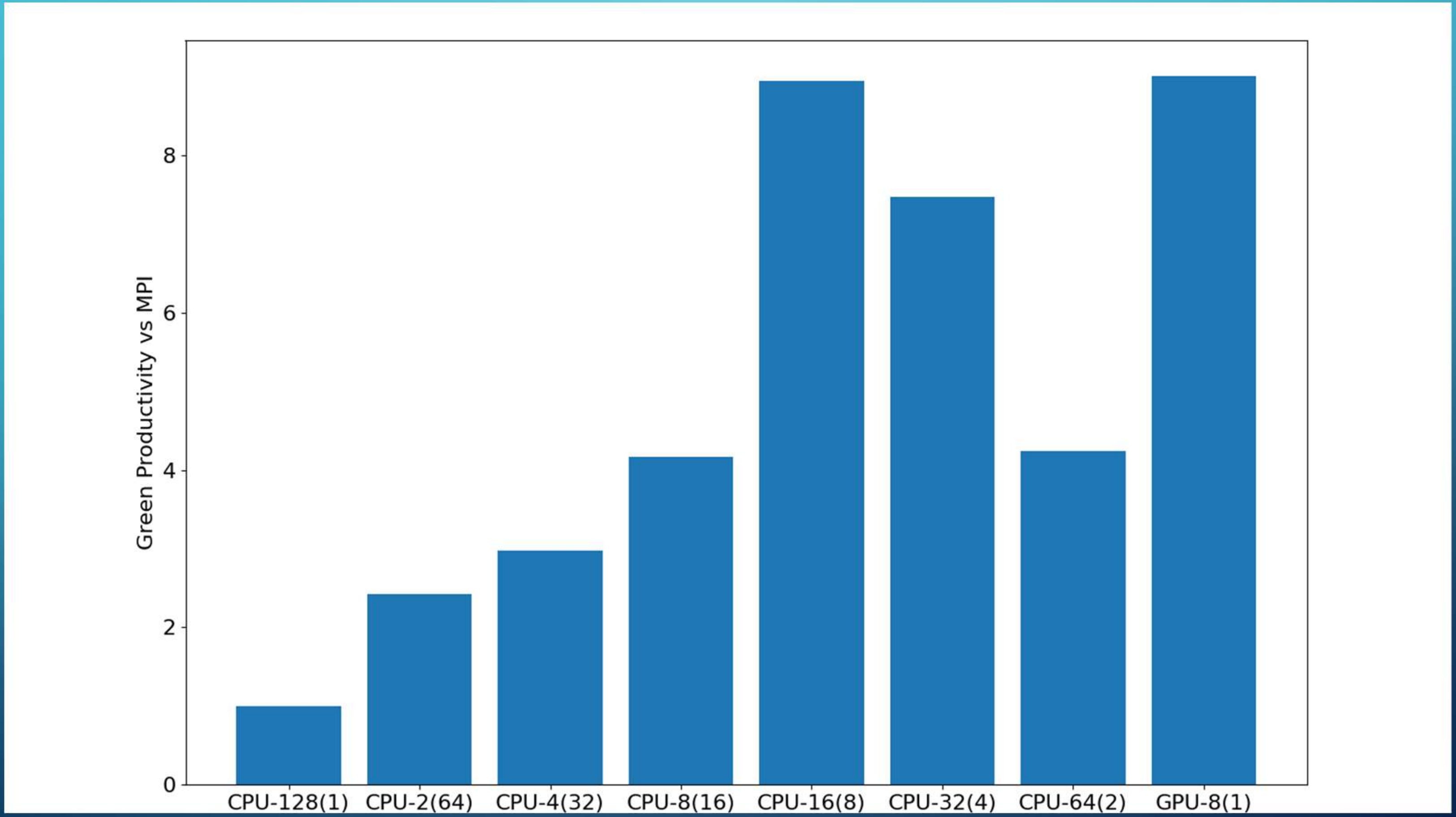
CONFIG	TASKS	THREADS	GPUS	NODES
HYBRID	2	64	0	1
HYBRID	4	32	0	1
HYBRID	8	16	0	1
HYBRID	16	8	0	1
HYBRID	32	4	0	1
HYBRID	64	2	0	1

CONFIG	TASKS	THREADS	GPUS	NODES
GPU	8	1	8	1

CLANG-16 COMPILER/CRAY-FFTW

Tasks(Threads)	Energy (KJ)	Total Time (sec)	I/O (Reading) (sec)	Gridding (sec)	Reduce (sec)	FFTW (sec)	Phase corr. (sec)
CPU 128(1)	45.6920 +- 1.9914	69.9672 +- 0.4265	0.4809 +- 0.0075	0.8032 +- 0.0042	58.6788 +- 0.2854	1.2107 +- 0.0288	0.2550 +- 0.0007
CPU 2(64)	26.7525 +- 0.6316	49.4272 +- 0.1804	2.9106 +- 0.0425	13.3607 +- 0.1627	4.6483 +- 0.4575	10.7353 +- 0.3677	0.9258 +- 0.0299
CPU 4(32)	23.6025 +- 0.3683	45.4316 +- 0.2830	2.5022 +- 0.0123	6.6939 +- 0.0439	3.3854 +- 0.0257	15.9419 +- 0.4023	0.9077 +- 0.0101
CPU 8(16)	19.5200 +- 1.1724	39.2875 +- 1.2176	2.3412 +- 0.0284	2.9990 +- 0.0141	3.8123 +- 0.0085	12.939 +- 1.0483	0.7066 +- 0.0304
CPU 16(8)	13.2875 +- 0.4298	26.8608 +- 0.5055	2.2616 +- 0.0543	1.6988 +- 0.0065	4.2890 +- 0.0122	1.9139 +- 0.0609	0.5021 +- 0.0292
CPU 32(4)	14.7575 +- 0.2767	28.9759 +- 0.5368	2.3161 +- 0.0196	1.1969 +- 0.0051	6.2361 +- 0.0065	1.3992 +- 0.0365	0.4967 +- 0.0174
CPU 64(2)	21.3600 +- 1.8238	35.3148 +- 0.3432	2.3574 +- 0.0299	0.9314 +- 0.0041	11.0904 +- 0.0169	1.2801 +- 0.0147	0.4565 +- 0.0006
GPU 8(1)	20.8350 +- 1.8109	17.0166 +- 0.4256	0.4628 +- 0.0016	1.0944 +- 0.0004	2.0018 +- 0.0176	11.0371 +- 0.2919	0.1107 +- 0.0001

CLANG-16 COMPILER/CRAY-FFTW



WHAT ABOUT THE I/O?

In RICK, I/O shows a huge variability in reading the measurement set from storage

WITH THE SAME CONFIGURATION READING ~80 GB FROM STORAGE CAN TAKE FROM 2 SECONDS UP TO 350 SECONDS ON ONE NODE!!!

FUTURE PERSPECTIVES

- Implementation of the ADIOS2 library in RICK to avoid the I/O bottleneck
- Investigating the possibility to use MGARD to obtain data compression in read
- Implementation of the ROCFFTMp in RICK when available
- FUNDAMENTAL BUT VERY HARD TO DO: Apply GPU compression before communications to diminish the Reduce impact

CONCLUSIONS

- In multi-node case you cannot avoid using GPUs
- In single-node cases the hybrid solution is the greenest one

THANK YOU FOR THE ATTENTION!!!