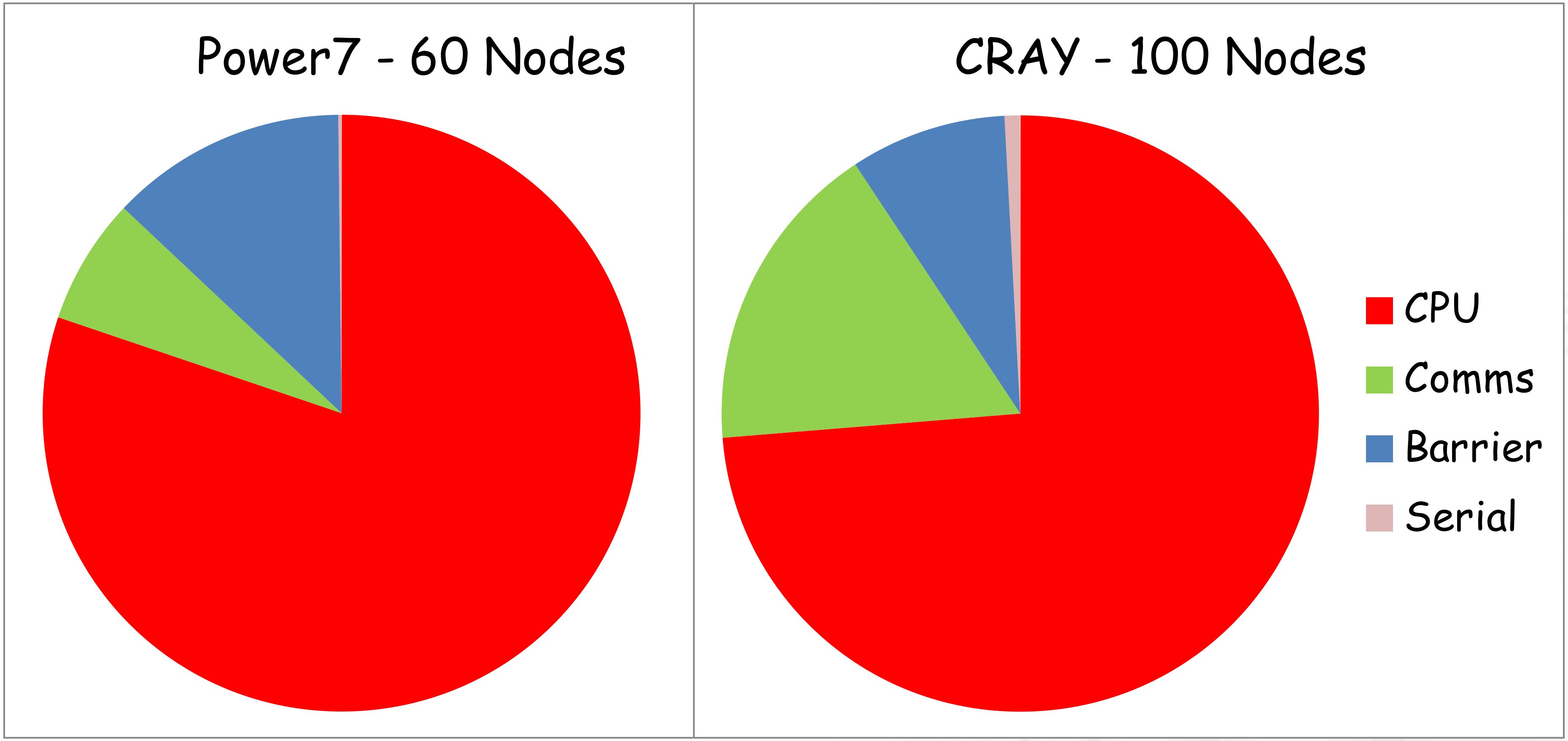
Streamlining HPC scenarios for future NWP

Sami.Saarinen@csc.fi (CSC – IT Center for Science Ltd, Finland)
with Deborah.Salmond@ecmwf.int (ECMWF)

April 14-15, 2014 for ECMWF Scalability workshop

PREVIOUSLY ON 24

IFS T1279L137 ~ 16km : 10-day FC : CY40R1



2258 seconds
5.1 Tflops (8.6% peak)

2182 seconds
5.2 Tflops (10.4% peak)

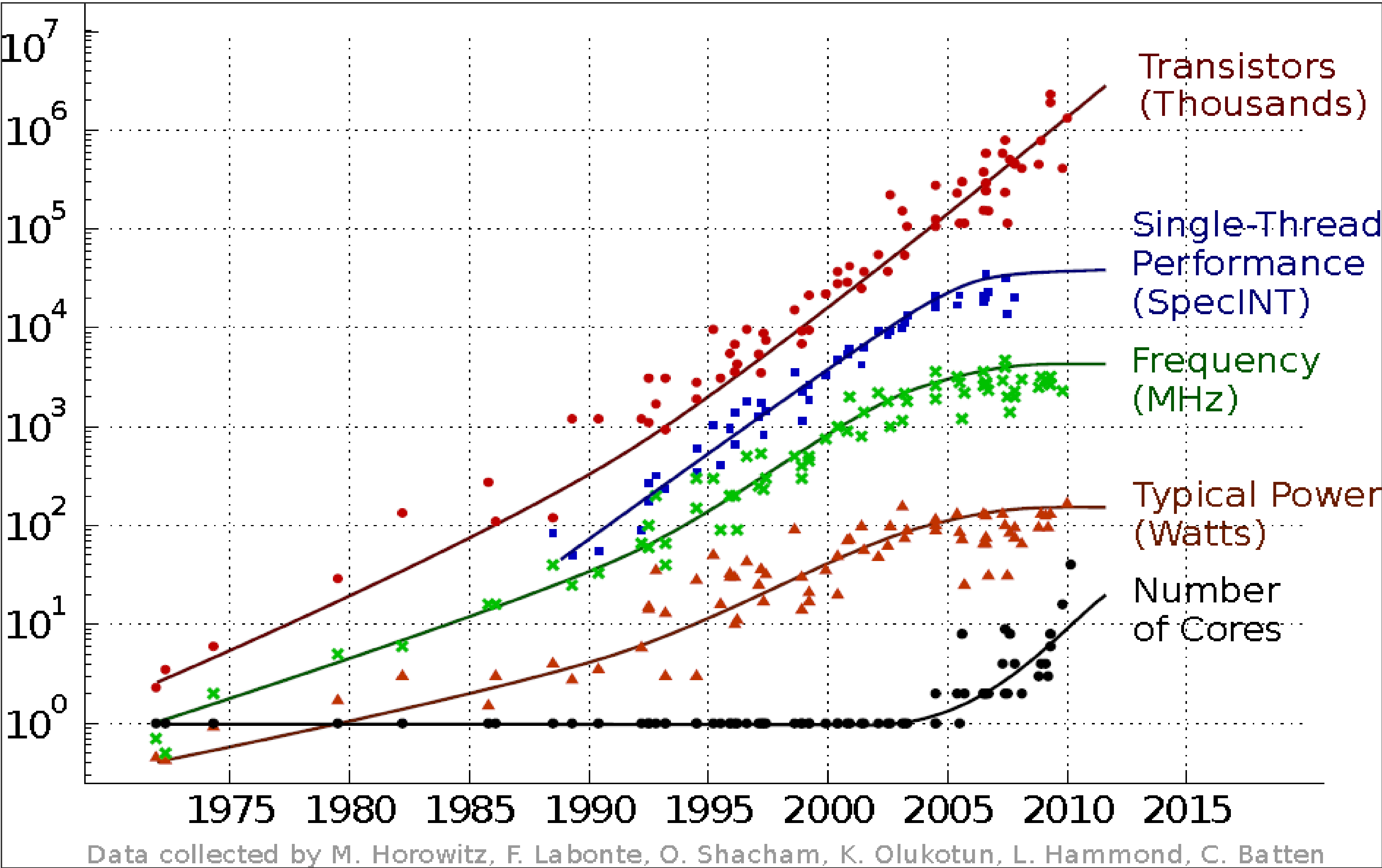
Outline of the talk

- Facts
- Scenarios
- Streamlining

FACTS

"A thing that is known or proved to be true"

The power wall



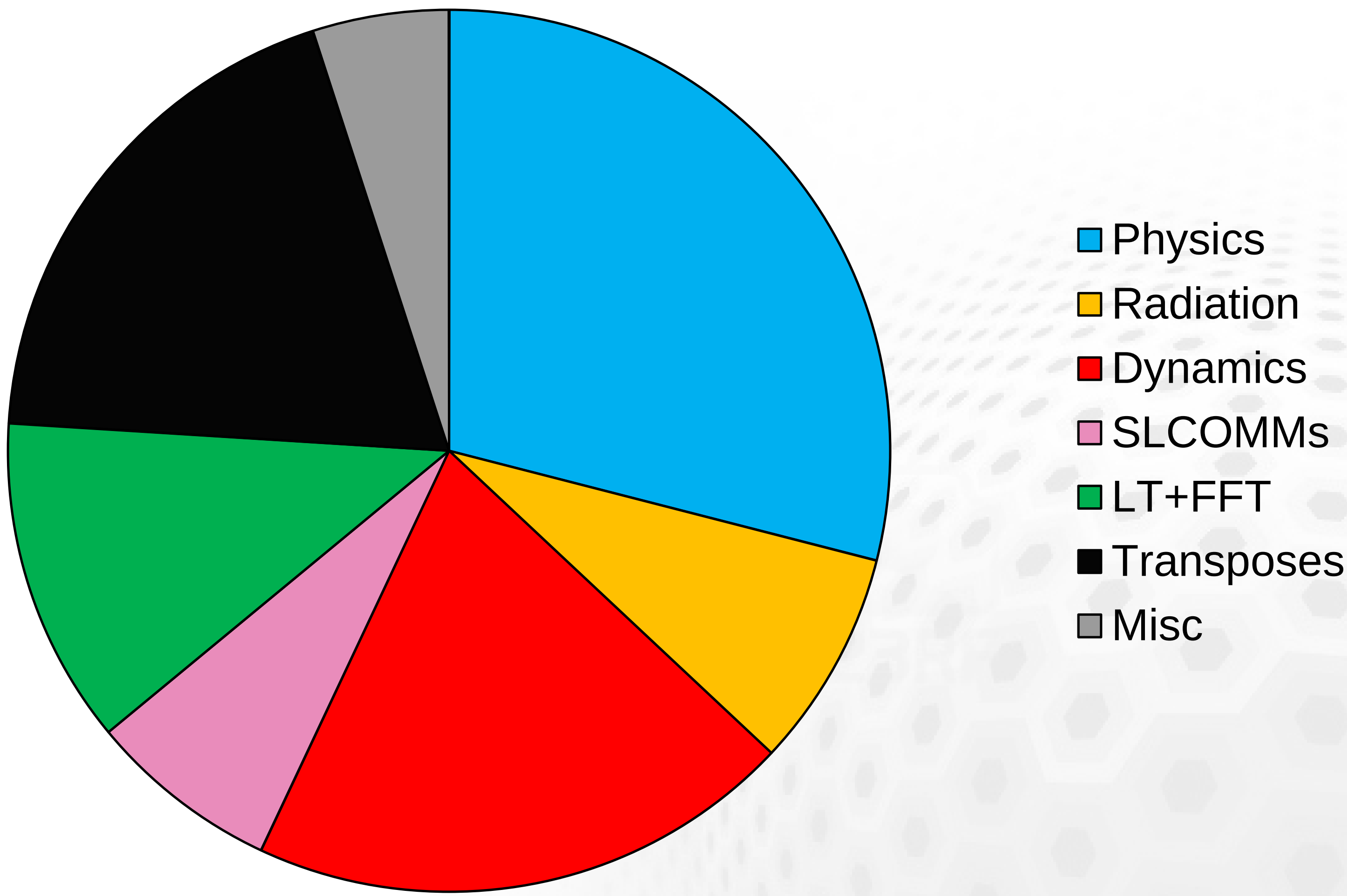
A "Total Recall" ?

- **CPU clock frequencies have practically ceased to increase about 10 years ago**
  – Power [W] ~ Freq^3 → lots of heat & €€€ (~ 1.4 $$$)
  – Frequencies ~ 1 … 3 GHz (except on IBM P6 @ 4.7GHz)
- **However, Moore's Law continues to be valid**
  – Requires increased investments in parallelism
  – GPUs and many-core techniques offer a viable option
- **A multi-objective optimization dilemma**
  – Power is capped by energy consumption limits
  – Yet much more computational performance is needed

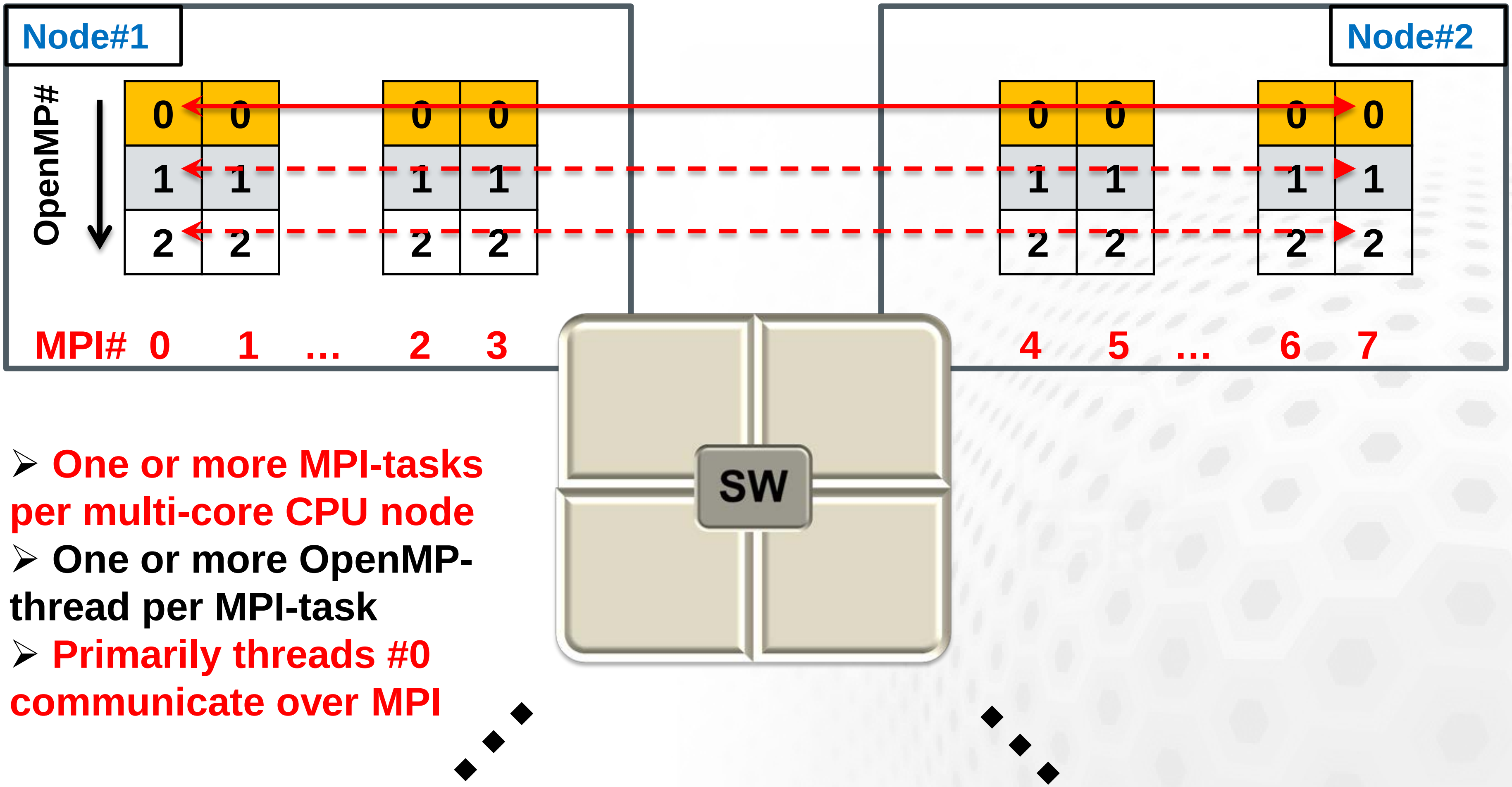
Targeting T2047L137 (~10km)

- ECMWF's near future operational FC model
- Sample performance data from Cray XC30 run
  - 128 nodes, 24-cores/node in 2 sockets, 64GB/node
  - Ivy Bridge E5-2697 v2 (2.7GHz) – TDP 130W/socket
- 10-day forecast : 1024 MPI x 6-way OpenMP
  - Compiled with Cray CCE 8.2.2 and uses 2-way HT
- Time step : 450s
- Total elapsed time : 6242s (~1h 44min)
- Baseline energy @ 90% TDP : 51.9 kWh

10-day T2047L137 ~ 10km
t = 6242s @ 51.9 kWh



IFS parallelization over MPI + OpenMP



- ➢ One or more MPI-tasks per multi-core CPU node
- ➢ One or more OpenMP-thread per MPI-task
- ➢ Primarily threads #0 communicate over MPI

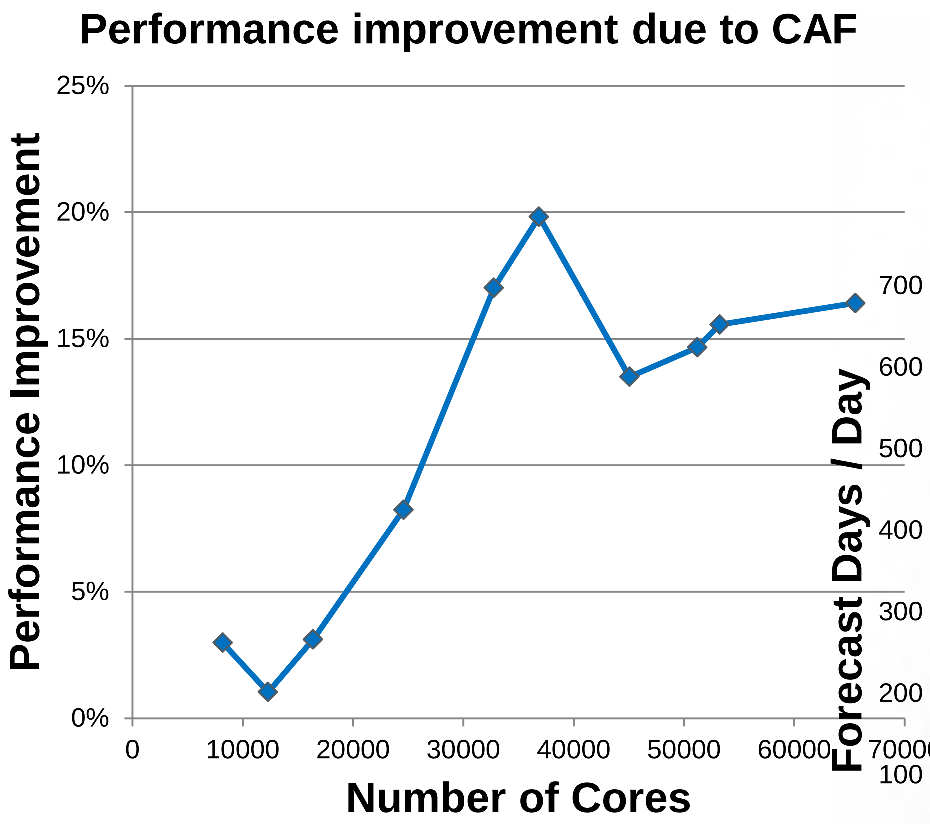
SCENARIOS

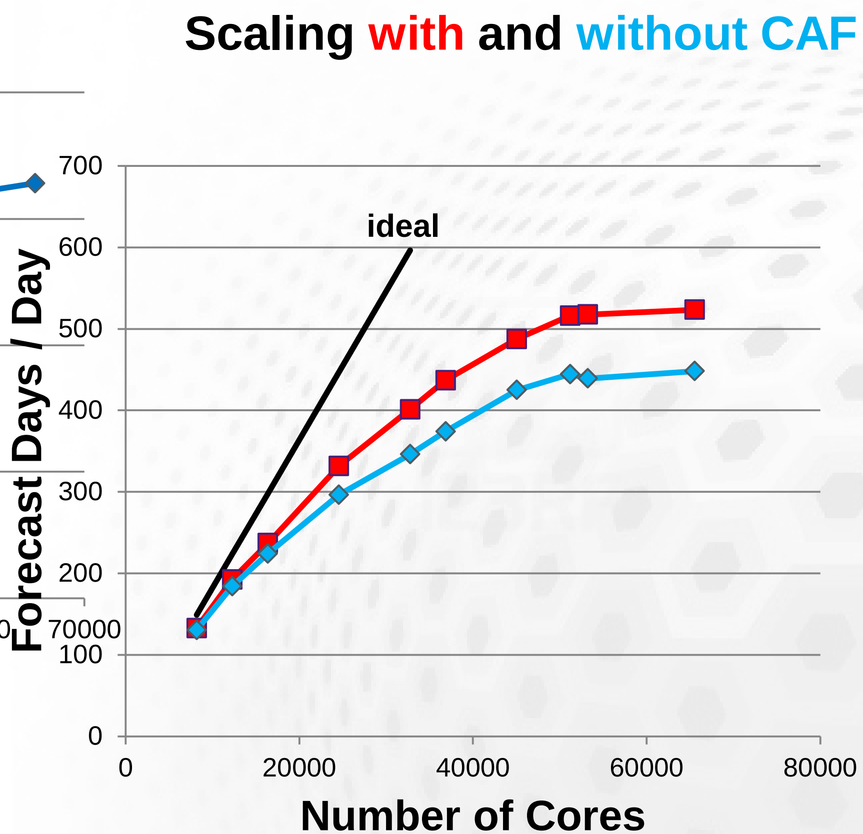
"A written outline of a film, novel, or stage work giving details of the plot and individual scenes"

On CPU-side CAF-scaling not too bad …

(T2047L137/RAPS12 CY37R3 on HECToR, Cray XE6)

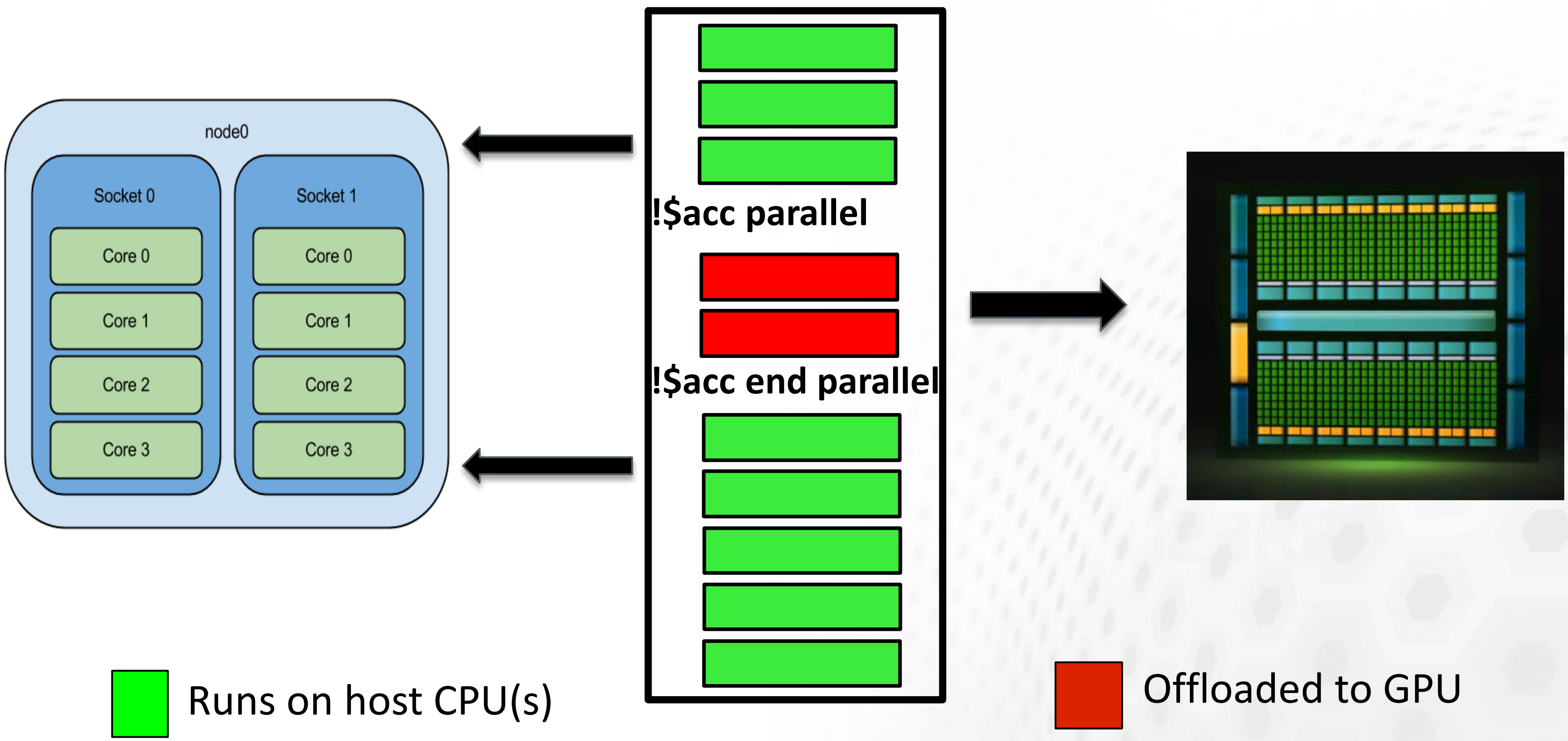
(Courtesy George Mozdzynski, ECMWF)





Using GPUs with help of OpenACC

IFS



DAXPY with OpenMP & OpenACC

```
SUBROUTINE daxpy(n, a, x, y)
INTEGER :: n, j
REAL(kind=8) :: a, x(n), y(n)
!$omp parallel do
  DO j = 1,n
     y(j) = y(j) + a * x(j)
  ENDDO
!$omp end parallel do
END SUBROUTINE daxpy

! call daxpy with 128M elements
CALL daxpy(2**27, 3.14_8, x, y);
```

```
SUBROUTINE daxpy(n, a, x, y)
INTEGER :: n, j
REAL(kind=8) :: a, x(n), y(n)
!$acc parallel loop
  DO j = 1,n
     y(j) = y(j) + a * x(j)
  ENDDO
!$acc end parallel loop
END SUBROUTINE daxpy

! call daxpy with 128M elements
CALL daxpy(2**27, 3.14_8, x, y);
```

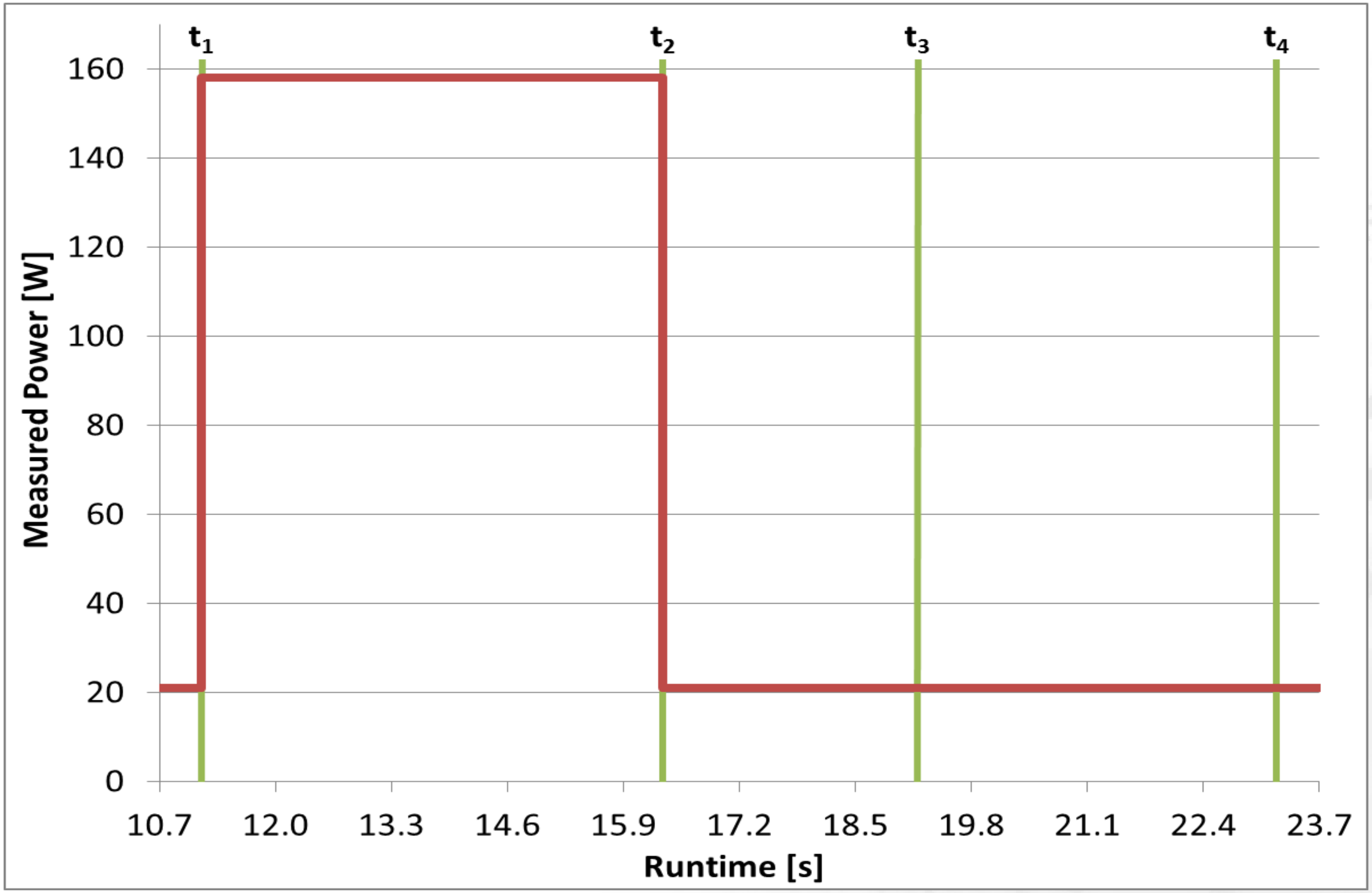
Tempted to go for GPUs ?

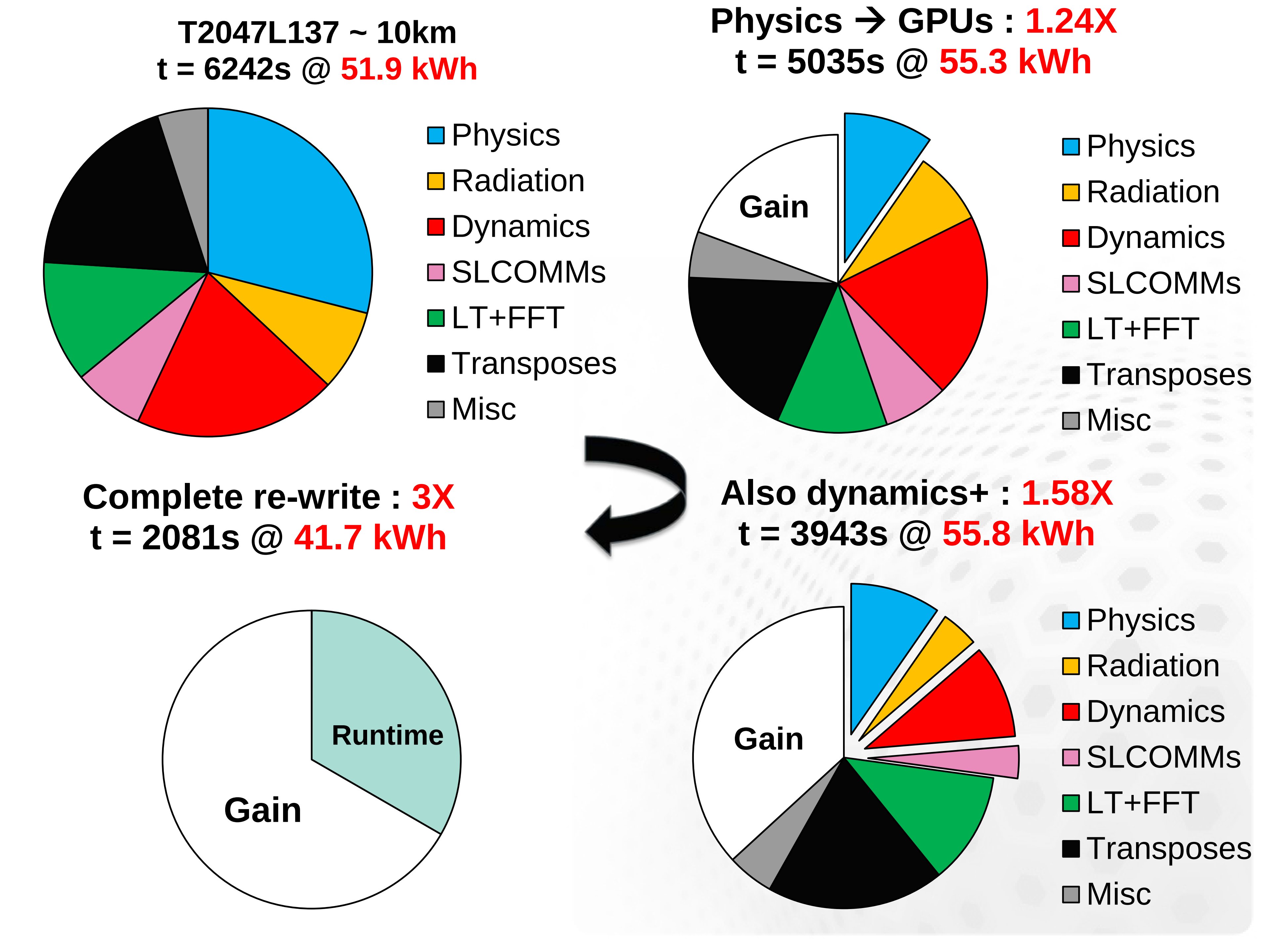
- **Lets perform a "back-of-an-envelope study"**
  - How well could IFS scale on GPUs ?
  - (When) Are we going to save in our energy bill ?
- **Speculating with T2047L137 on CPUs+GPUs**
  - Physics (~29%) to GPUs → target 3X speedup here
  - Plus most of dynamics (~35%) with speedup of 2X
  - Complete code re-write with total speedup of 3X

- **Assume 2 x Kepler K40 (12GB) per IvB-node**
  - Total 256 K40 GPUs with GDR MPI + Hyper-Q/MPS
  - TDP value 235W (~70% will be used), idle ~20W

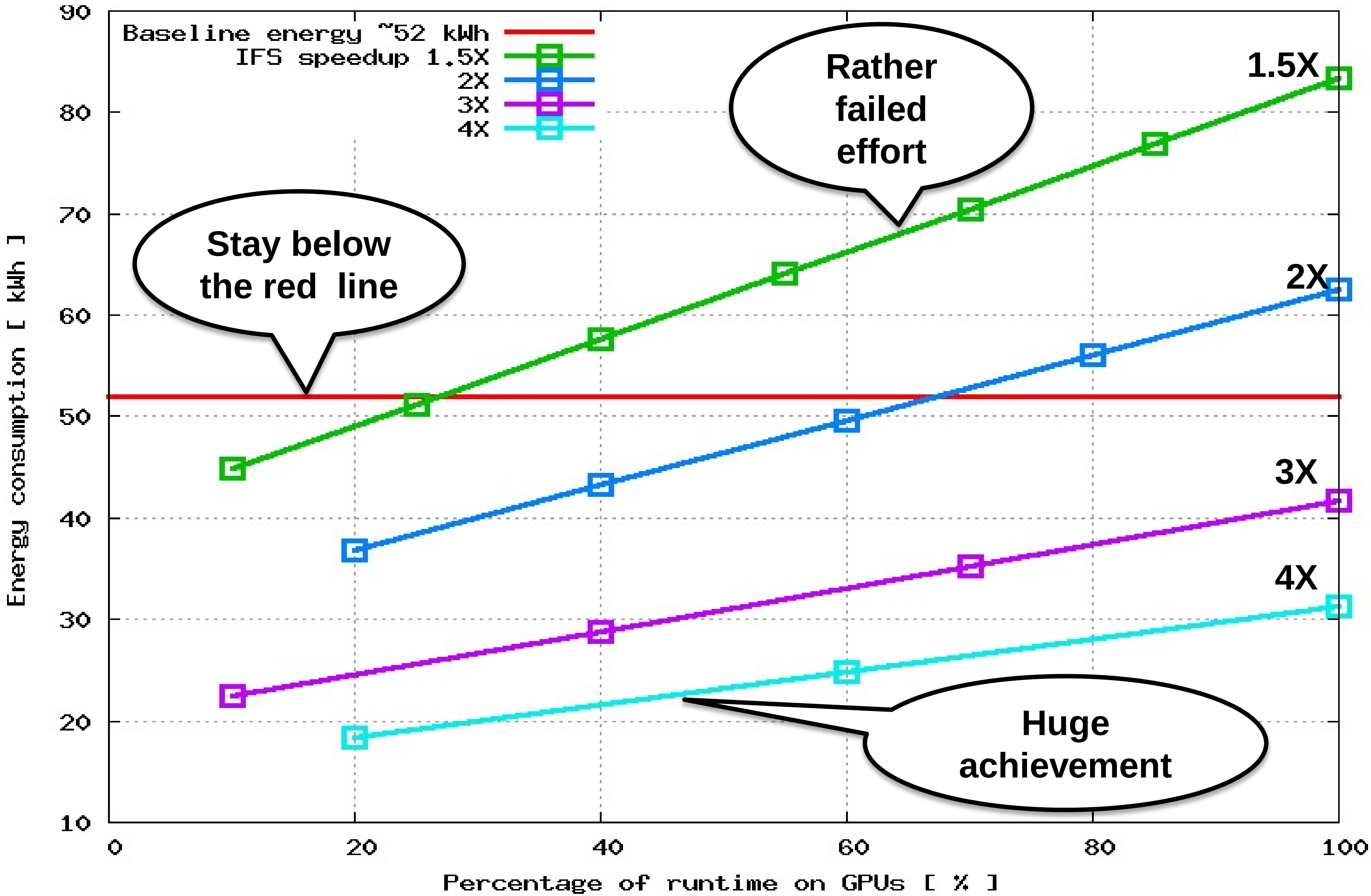
Expected power [W] profile on GPUs

(Courtesy Martin Burtscher, TX State Univ)

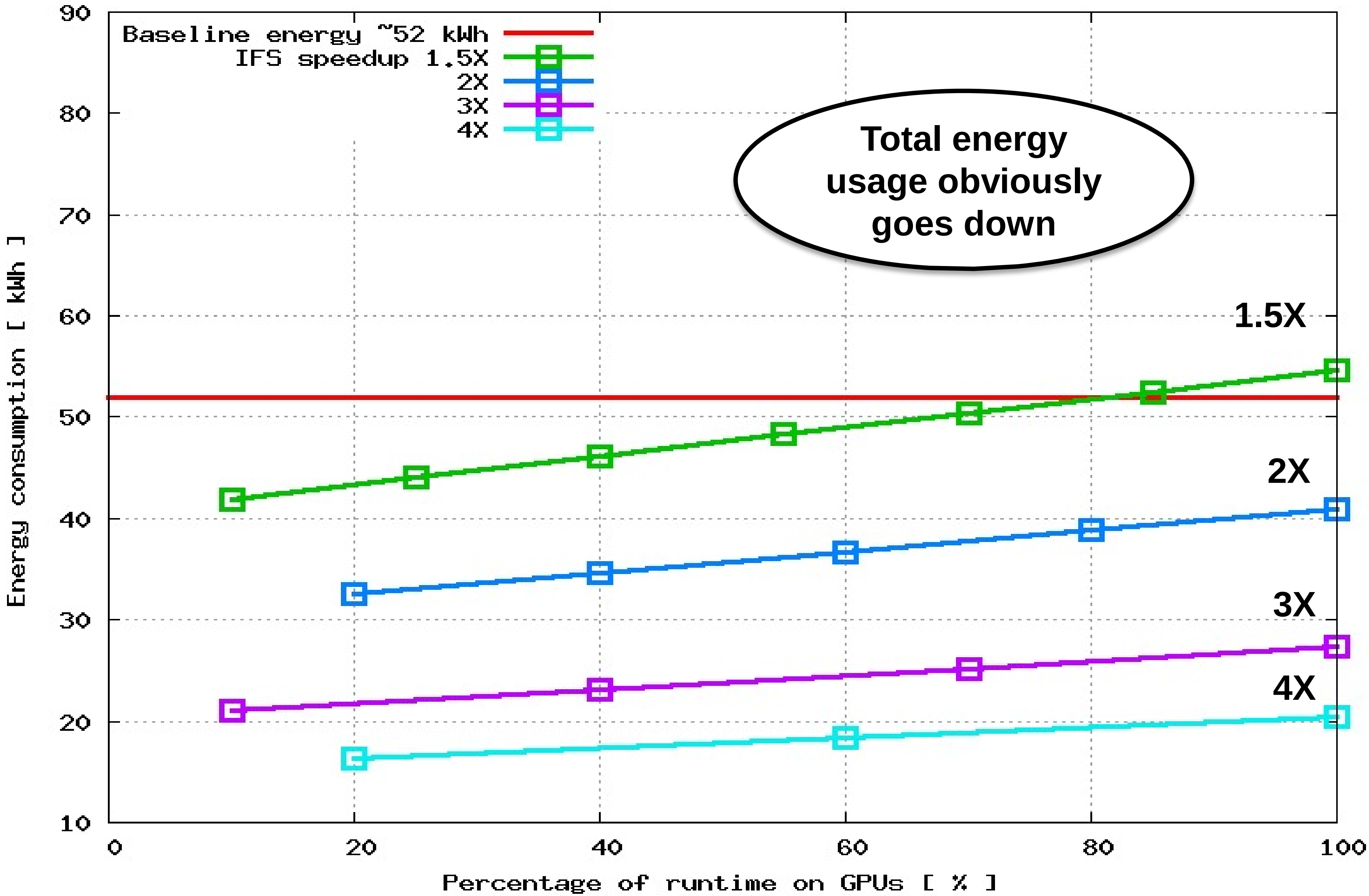



T2047L137 ~ 10km
t = 6242s @ 51.9 kWh
Physics
Radiation
Dynamics
SLCOMMs
LT+FFT
Transposes
Misc
Physics → GPUs : 1.24X
t = 5035s @ 55.3 kWh
Gain
Physics
Radiation
Dynamics
SLCOMMs
LT+FFT
Transposes
Misc
Complete re-write : 3X
t = 2081s @ 41.7 kWh
Runtime
Gain
Also dynamics+ : 1.58X
t = 3943s @ 55.8 kWh
Gain
Physics
Radiation
Dynamics
SLCOMMs
LT+FFT
Transposes
Misc

Energy vs. IFS speedup vs. GPU-%



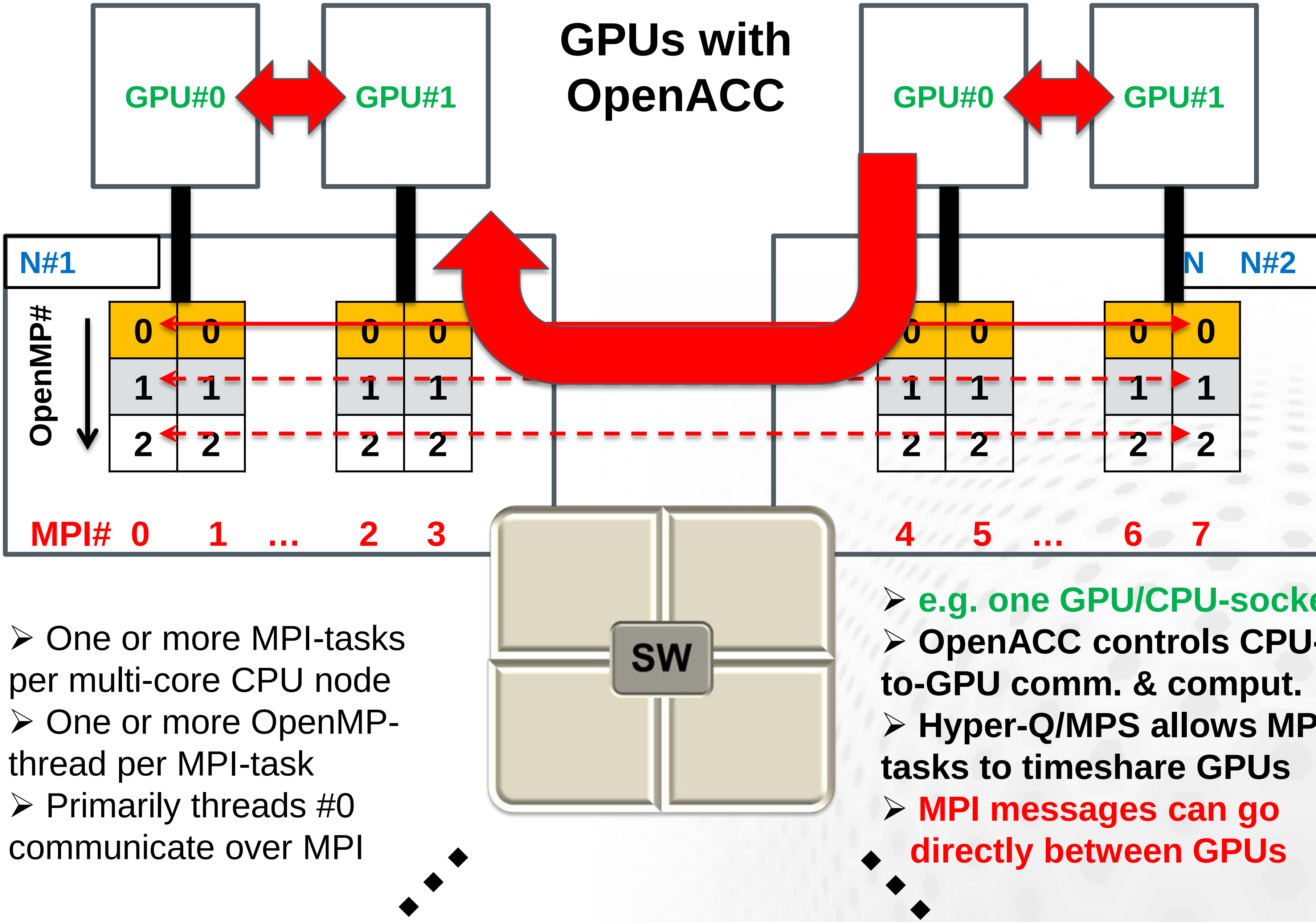
Allow CPUs ~ idle when on GPU regions



STREAMLINING

"Make (an organization or system) more efficient and effective by employing faster or simpler working methods"

GPUs with OpenACC



- One or more MPI-tasks per multi-core CPU node
- One or more OpenMP-thread per MPI-task
- Primarily threads #0 communicate over MPI

- **e.g. one GPU/CPU-socket**
- **OpenACC controls CPU-to-GPU comm. & comput.**
- **Hyper-Q/MPS allows MPI-tasks to timeshare GPUs**
- **MPI messages can go directly between GPUs**

Streamlining suggestions [1]

- **Look at OpenMP regions ~ OpenACC ”friendly”**
  – Start from physics – usually no MPI involved
- **Create data on GPUs and try to keep it there**
  – Minimize transfers between host CPUs
- **Optimize with CUDA – call it from OpenACC**
  – Use high performance CUDA-libraries
- **Use all allowable asynchronous operations with OpenACC – GPUs like to *“drink from a hosepipe”***
  – Feed GPUs with more data whilst previous computed

Streamlining suggestions [2]

- **Direct MPI-link between GPU-to-GPU exists**
  - Direct device resident data exchange between GPUs
- **Simplify some MPI coding on CPUs with CAF**
  - *Caveat* : depends heavily on use of Cray compiler …
- **Remember: energy savings eventually reachable**
  - When the major part of code runs on GPUs switch also to less energy consuming CPUs – saves you some £££'s
- ***But* : without a major code restructuring and algorithmic changes good computational performance & energy efficiency difficult to obtain**

Compiler support for accelerated computing as of 1Q/2014

	Cray	Intel	PGI	GNU	CAPS
OpenACC (GPUs)	Yes	No	Yes	2015?	Yes
OpenACC (MICs)		No			Yes
OpenMP 4.0 (MICs)	Soon	Yes	No	??	
CAF	Yes	Without MPI	No	??	
CUDA (nvcc)	(Yes)		(Yes)		(Yes)
CUDA Fortran			Yes		

Acknowledgements

- Prof. Martin Burtscher, Texas State University, for exhilarating discussions & learning material on how to calculate energy consumption
- Peter Towers for providing T2047L137 data
- George Mozdzynski for CAF material
- Peter Messmer from nVidia for encouraging discussions and excellent CUDA teaching
- And finally Olli-Pekka Lehto & Tommi Tervo from CSC for interpreting the power figures

Some references

- Herb Sutter : *”The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software”* , http://www.gotw.ca , DDJ 3/2005
- Martin Burtscher : *“Accurate Power and Energy Measurement on Kepler-based Tesla GPUs”* , GTC2014, San Jose, CA
- X.Lapillonne, O.Fuhrer : *”Using compiler directives to port large scientific applications to GPUs: An example from atmospheric science”* , 2/2014
- George Mozdzynski : *”IFS Optimisations for ExaScale & Co-design”* , CRESTA 3rd Collaboration Meeting, Stockholm, 9/2012

Vocabulary

- CAF = Co-Array Fortran (part of Fortran 2008)
- CUDA = Compute Unified Device Architecture
- GDR = GPUDirect RDMA allows exchange of GPU-data directly between MPI-tasks
- GPU = Graphics Processing Unit
- Hyper-Q = Allows CUDA kernels to be processed concurrently on the same GPU
- MPS = Multi-Process Service allows sharing a GPU between multiple MPI-tasks
- RDMA = Remote Direct Memory Access
- TDP = Thermal Design Power