# Energy Simulation with SimGrid

Millian Poquet

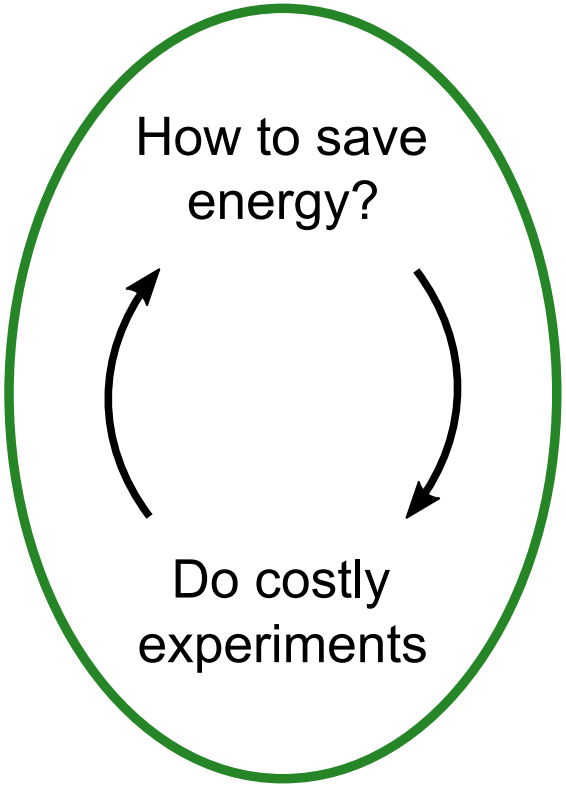
millian.poquet@inria.fr

Slides from SimGrid tutorials and F. C. Heinrich (Cluster'17)






# Chicken-and-egg Situation




- Typically: MJ to save some %
- Classical issue in optimization...

Can we do more reasonable experiments?

# Simulation at rescue

**The fastest path from idea to data.**

Comfortable

- Thousands of runs within the week on your laptop
- Preliminary results from partial implementations
- Focus on ideas, don't fiddle with technical subtleties (yet)

Challenges

- **Validity**: Realistic results (controlled experimental bias)
- **Scalability**: Simulate *big enough* problems *fast enough*
- **Applicability**: Should simulate what is important to users

# Outline

1 Introduction

2 Overview and Models

3 Validation (CLUSTER'17)

4 Conclusion

# SimGrid at a glance

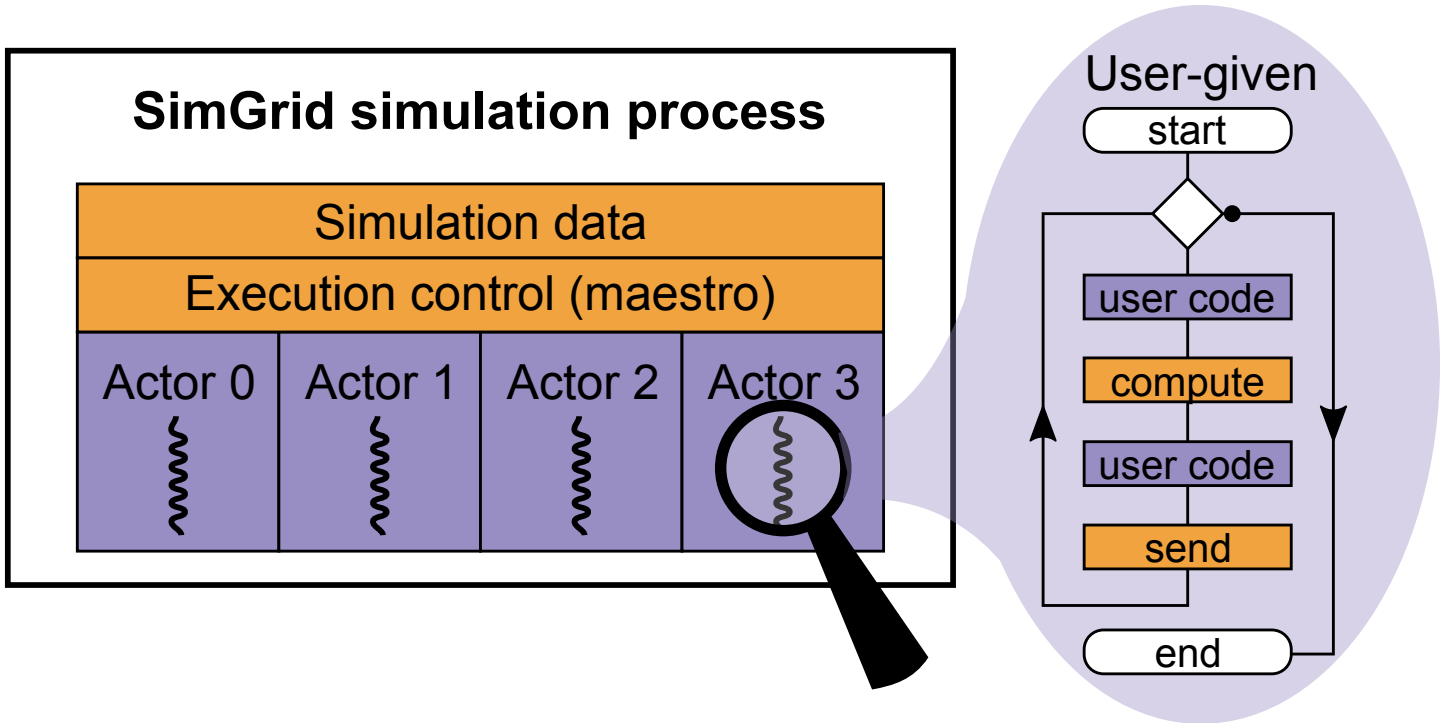
- **18**-year-old open-source project
- Collaboration: **France** (Inria, CNRS, Grenoble, Lyon, Rennes...), **US** (UCSD, Hawaii), UK, Austria (Vienna)...
- Papers: 500 cite, 300 use, 60 extend
- LOC: ≈150k C/C++

- Initially focused on Grids. Argue that same techniques can be used for P2P, HPC, Cloud...
- Goal: **Usable** tool with **predictive** capability
- Model Checking capabilities

# Software Architecture

Essentially a library. Architectured as an OS.

- 1 *system* process (kernel + user code)
- mutual exclusion on actors' execution
- *maestro* dictates who run
- user code increases simulation time via syscalls



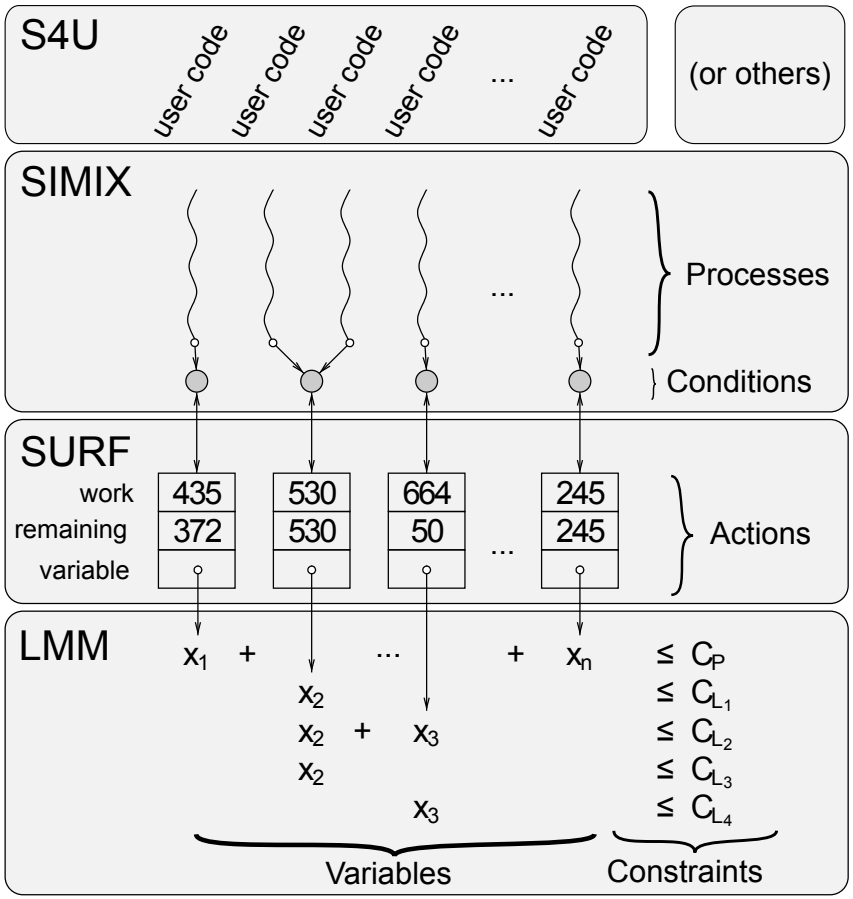
# Internals Organization

User-visible components

- S4U (MSG): general purpose
- SimDag: DAGs of ptasks
- SMPI: online/offline MPI

Internally: Strict layers

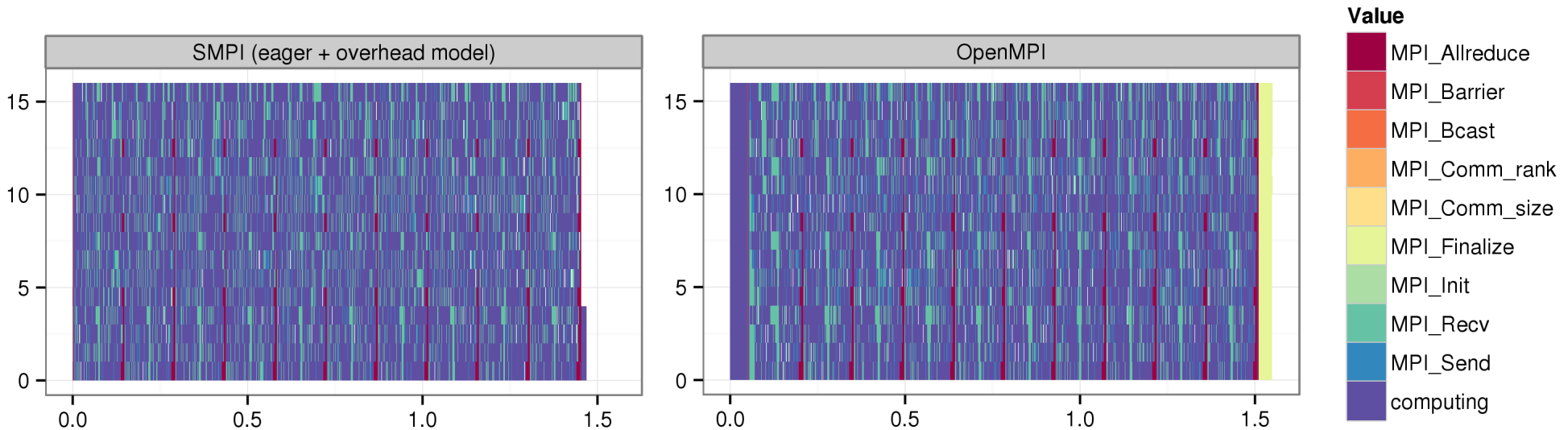
- S4U: User-friendly sugar
- SIMIX: Processes, synchro
- SURF: Resources usage
- Models: Action completion computation



# Network Models

Several are available:

- Fast flow-based, towards realism and speed (by default)
  Contention, slow start, TCP congestion, cross-traffic effects.
- Constant time: A bit faster, no hope for realism
- Coordinate-based: Easier to instantiate P2P scenarios
- Packet-level: NS3 bindings



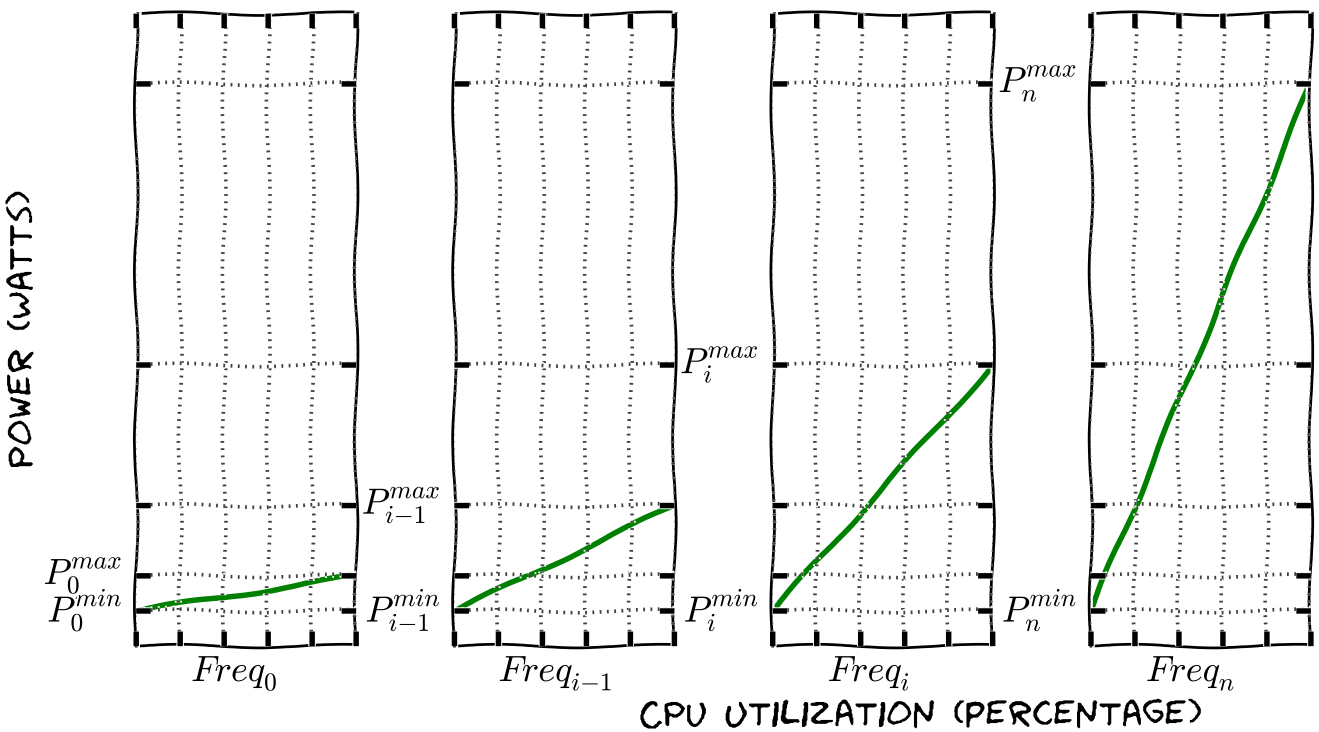
# DVFS and Energy Model

DVFS

- Modern CPUs can reduce computation speed to save energy
- Power states: Levels of performance. *Governors* pick them.
- SimGrid: Manually switch pstates, which change the flop rate

Energy Model

- For one pstate, consumption = linear function of CPU use
- Classically accepted model in the literature, rarely challenged



# Basic Energy Model Instantiation

```
<host id="MyHost2" speed="100.0Mf" >
   <prop id="watt_per_state" value="100.0:200.0" />
   <prop id="watt_off" value="10" />
</host>
```

- `watt_off`: the host is off $\Longrightarrow$ 10 Watts
- `watt_per_state` power consumption interval [min:max]
  - Idling host $\Longrightarrow$ 100 Watts
  - Fully loaded host (`100.0Mf`=100 MFlops/s) $\Longrightarrow$ 200 Watts
  - Linear model in between: CPU loaded at 50% $\Longrightarrow$ 150 Watts

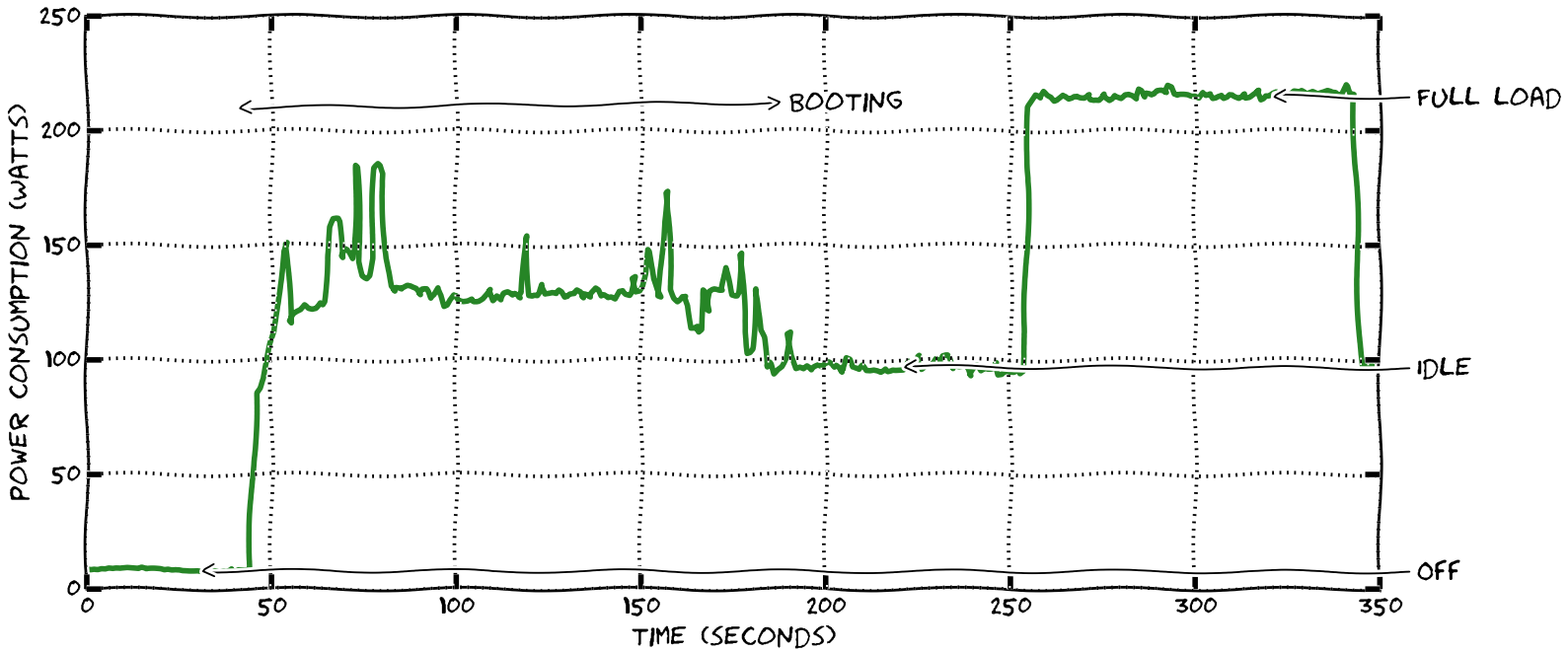
# DVFS Energy Model Instantiation

```
<host id="MyHost1" speed="100.0Mf,50.0Mf,20.0Mf" pstate="0" >
   <prop id="watt_per_state"
         value="95.0:200.0, 93.0:170.0, 90.0:150.0" />
   <prop id="watt_off" value="10" />
</host>
```

- `power`: 3 pstates $\{0, 1, 2\}$: 100, 50 and 20 Mflops/s
- `pstate`: Initial pstate (here, pstate=0, ie. 100 Mflops/s)
- `watt_per_state` two power values [min:max] as before
  - Here, CPU loaded at 50% in pstate 2 consumes 120 Watts.
  - Remember, pstates are numbered from 0!
    pstate 2 is 20 Mflops/s peak

# ON/OFF Energy Model

ON ↔ OFF takes time (seconds) and energy (Joules).




Many ways to do it

- Not easy for the noise: everybody wants something specific
- SimGrid provides basic mechanisms, you have to help yourself
- Switching on/off is instantaneous

# CLUSTER'17 paper

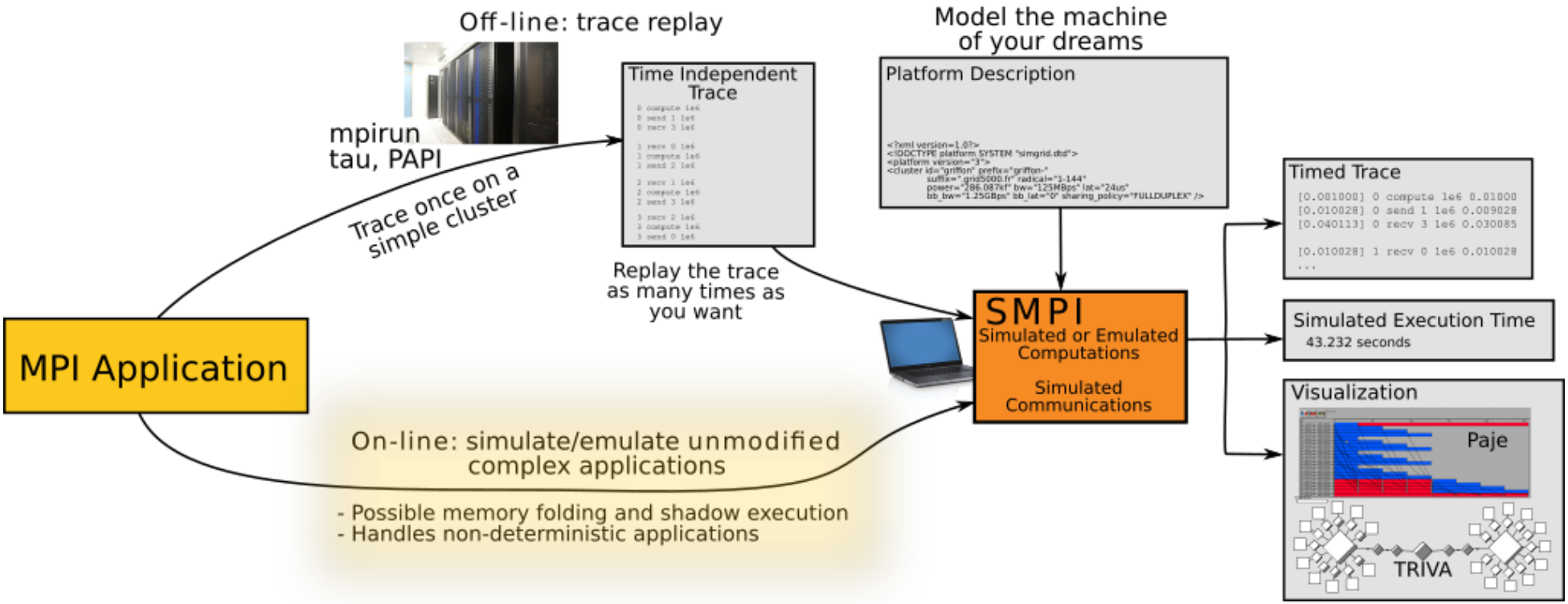
Heinrich, Cornebize, Degomme, Legrand, Carpen-Amarie, Hunold, Orgerie, Quinson: *Predicting the Energy-Consumption of MPI Applications at Scale Using Only a Single Node.*

Main goal: Validate performance and energy predictions

Quick overview:

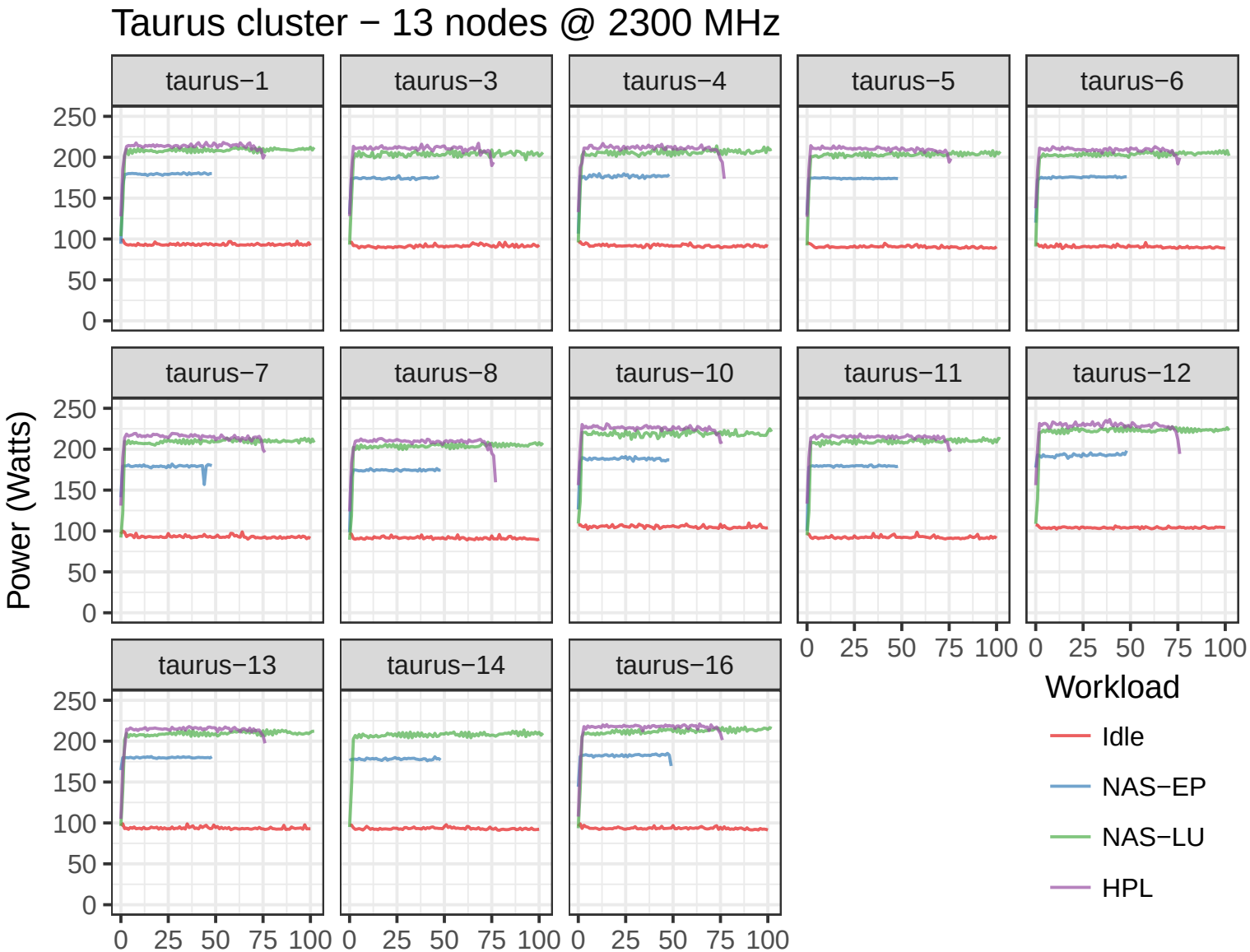
1. Obtain a platform model
   - How does MPI perform on **this** platform?
2. Run the application on one node, all cores
   - Processes interferences (memory contention, L1-L3 caches)
   - Measure the energy consumption
3. Run the application on one node, one core
   - Measure the energy consumption
4. Feed measurements / platform model into simulator

# MPI Simulation in SimGrid



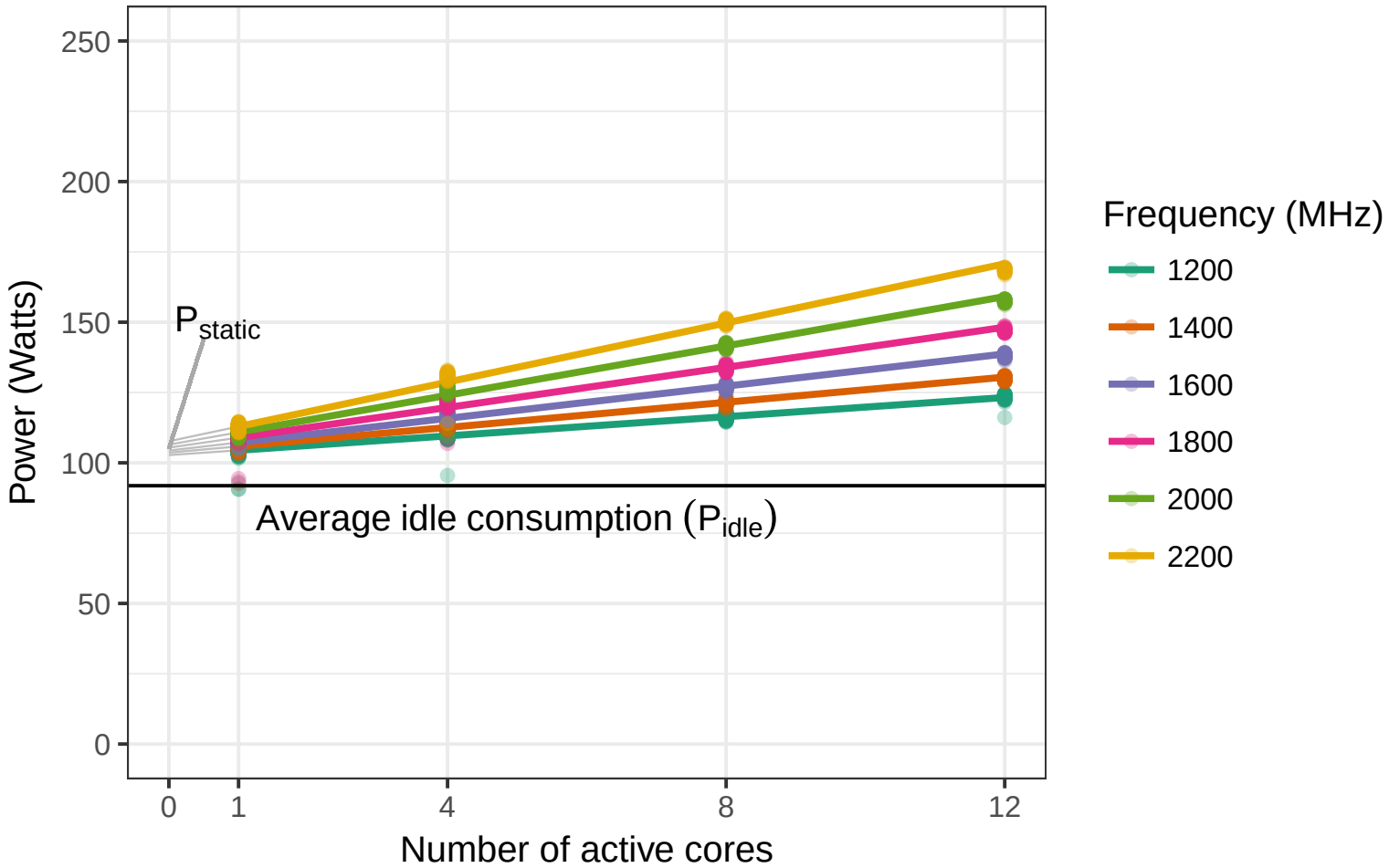
# Contribution 1: Problem

Energy Model should be **application-dependent**.

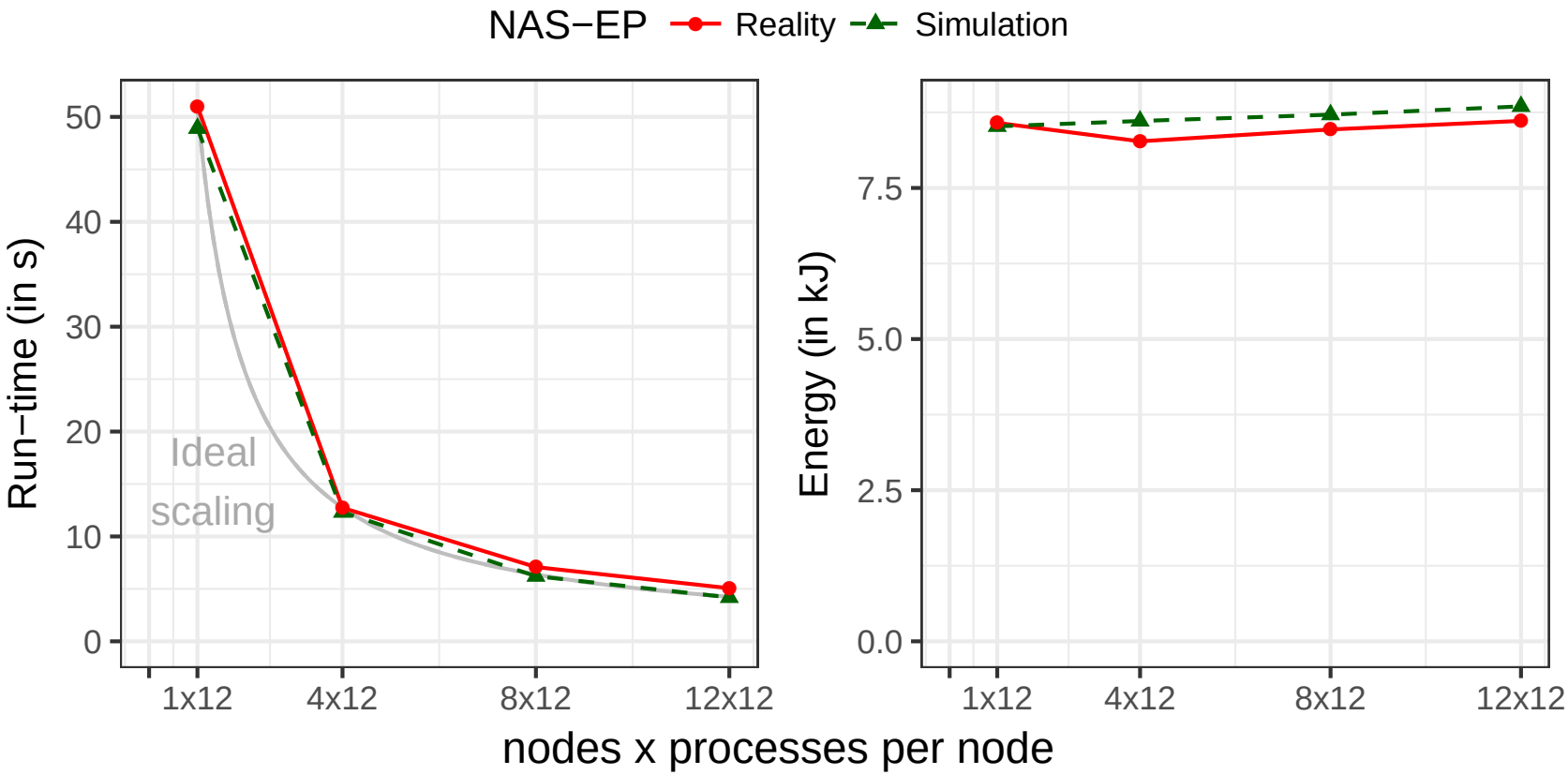


# Contribution 1: Solution

Instantiate the energy model presented before!

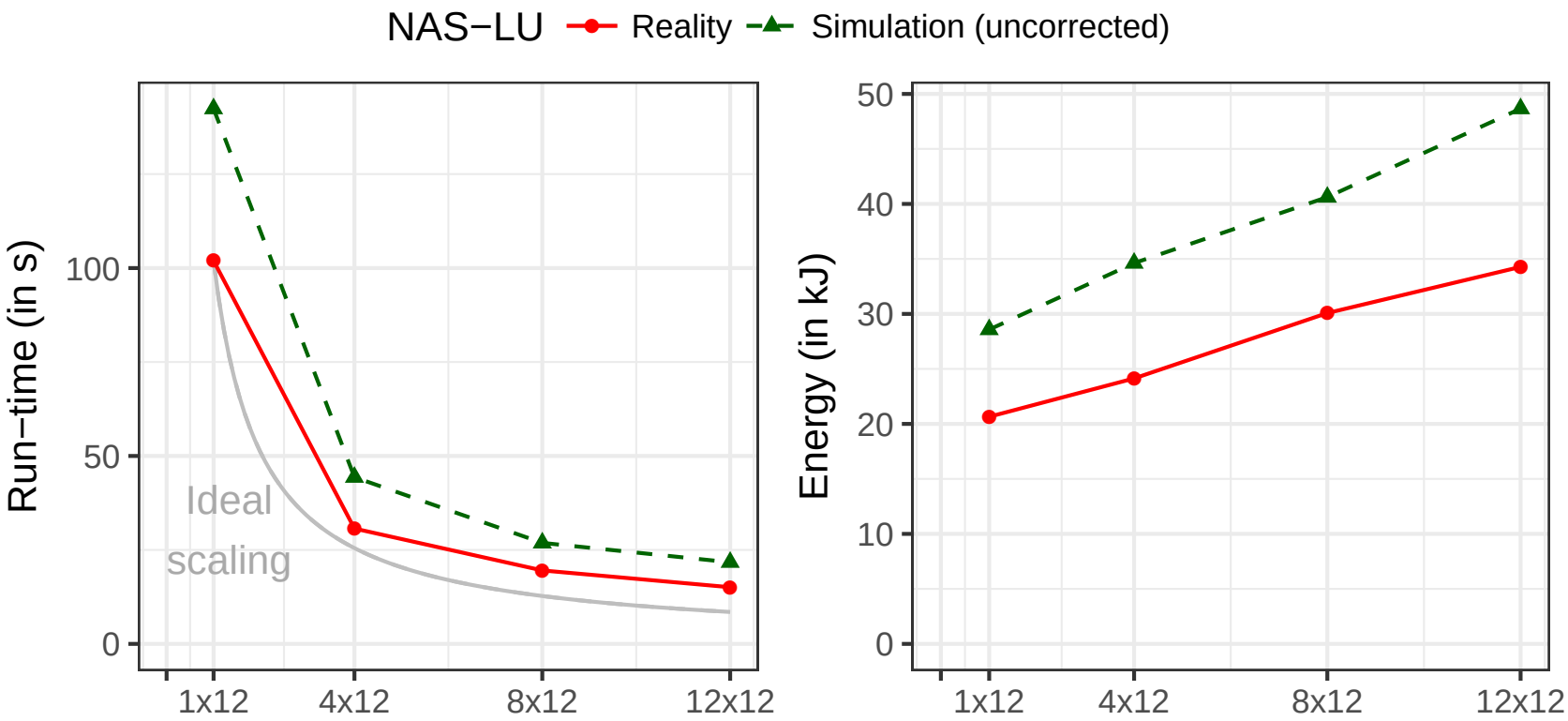


# Contribution 1: Outcome



# Contribution 2: Problem

- Previous benchmark (NAS-EP) uses **almost no communication**. What about more complicated applications?
- **NAS-LU** uses collective communciations and is memory bound
- Applications often contend e.g., on L1 or L3 caches



# Contribution 2: Solution

We unbias by computing speedup factors through trace alignment.

```
28  ...
    Region 43 if( iex .eq. 0 ) then
30  Region 43          if( north .ne. -1 ) then
                          call MPI_RECV( dum1(1,jst),
32              >                         5*(jend-jst+1),
                >                         dp_type,
34              >                         north,
                >                         from_n,
36              >                         MPI_COMM_WORLD,
                >                         status,
38              >                         IERROR )
    Region 2              do j=jst,jend
40  Region 2                 g(1,0,j,k) = dum1(1,j)
    Region 2                 g(2,0,j,k) = dum1(2,j)
42  Region 2                 g(3,0,j,k) = dum1(3,j)
    Region 2                 g(4,0,j,k) = dum1(4,j)
44  Region 2                 g(5,0,j,k) = dum1(5,j)
    Region 2              enddo
46  Region 2          endif
    Region 2
48  Region 2          if( west .ne. -1 ) then
                          call MPI_RECV( dum1(1,ist),
50              >                         5*(iend-ist+1),
                >                         dp_type,
52              >                         west,
                >                         from_w,
54              >                         MPI_COMM_WORLD,
                >                         status,
56              >                         IERROR )
    Region 3              do i=ist,iend
58  Region 3                 g(1,i,0,k) = dum1(1,i)
    Region 3                 g(2,i,0,k) = dum1(2,i)
60  Region 3                 g(3,i,0,k) = dum1(3,i)
    Region 3                 g(4,i,0,k) = dum1(4,i)
62  Region 3                 g(5,i,0,k) = dum1(5,i)
    Region 3              enddo
64  Region 3          endif
    ...
```

Calibration$^{RL}$ trace (MPI)

| rank | start (s) | duration (mus) | state |
|---|---|---|---|
| ... | ... | ... | ... |
| 1 | 1.643388 | 1293 | mpi_allreduce |
| 1 | 1.644681 | 62 | Computing |
| 1 | 1.644743 | 82 | mpi_barrier |
| 1 | 1.644825 | 6454 | Computing |
| 1 | 1.651279 | 549 | mpi_recv |
| 1 | 1.651828 | 474 | Computing |
| 1 | 1.652302 | 53 | mpi_send |
| 1 | 1.652355 | 2 | Computing |
| 1 | 1.652357 | 15 | mpi_send |
| 1 | 1.652372 | 359 | Computing |
| 1 | 1.652731 | 11 | mpi_recv |
| 1 | 1.652742 | 462 | Computing |
| 1 | 1.653204 | 15 | mpi_send |
| 1 | 1.653219 | 1 | Computing |
| 1 | 1.653220 | 9 | mpi_send |
| 1 | 1.653229 | 376 | Computing |
| 1 | 1.653605 | 22 | mpi_recv |
| 1 | 1.653627 | 465 | Computing |
| 1 | 1.654092 | 16 | mpi_send |
| 1 | 1.654108 | 1 | Computing |
| ... | ... | ... | ... |

Calibration$^{SMPI}$ trace (uncorrected SMPI)

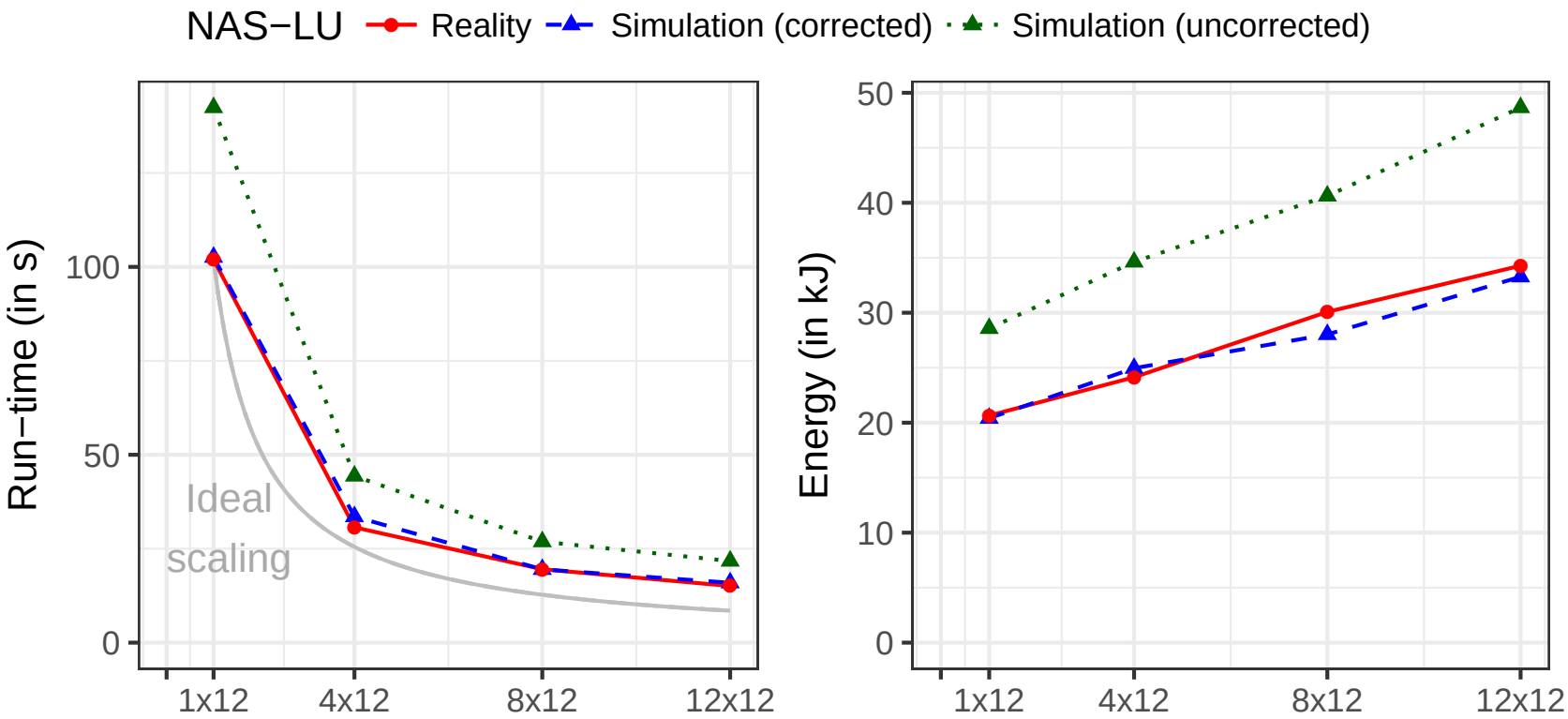
| start (s) | duration (mus) | state | Filename | Line |
|---|---|---|---|---|
| ... | ... | ... | ... | ... |
| 0.550426 | 1130 | mpi_allreduce | l2norm.f | 57 |
| 0.551556 | 18 | Computing | | |
| 0.551574 | 47 | mpi_barrier | ssor.f | 74 |
| 0.551621 | 5303 | Computing | | |
| 0.556924 | 617 | mpi_recv | exchange_1.f | 30 |
| 0.557541 | 608 | Computing | Region 3 | |
| 0.558149 | 4 | mpi_send | exchange_1.f | 113 |
| 0.558153 | 12 | Computing | Region 17 | |
| 0.558165 | 4 | mpi_send | exchange_1.f | 130 |
| 0.558169 | 652 | Computing | Region 18 | |
| 0.558821 | 8 | mpi_recv | exchange_1.f | 30 |
| 0.558829 | 587 | Computing | Region 3 | |
| 0.559416 | 5 | mpi_send | exchange_1.f | 113 |
| 0.559421 | 12 | Computing | Region 17 | |
| 0.559433 | 5 | mpi_send | exchange_1.f | 130 |
| 0.559438 | 699 | Computing | Region 18 | |
| 0.560137 | 9 | mpi_recv | exchange_1.f | 30 |
| 0.560146 | 597 | Computing | Region 3 | |
| 0.560743 | 4 | mpi_send | exchange_1.f | 113 |
| 0.560747 | 14 | Computing | Region 18 | |
| ... | ... | ... | ... | ... |

Merging traces

Region-based speedup/slowdown factors

```
"bcast_inputs.f:37:exchange_3.f:42",0.1655      Region 1
"exchange_1.f:30:exchange_1.f:48",14.6704       Region 2
"exchange_1.f:30:exchange_1.f:113",1.2967       Region 3
"exchange_1.f:30:exchange_1.f:130",1.2994       Region 4
. . .
"exchange_1.f:113:exchange_1.f:130",11.7101     Region 17
"exchange_1.f:130:exchange_1.f:30",1.9696       Region 18
...
"exchange_3.f:288:exchange_1.f:30",0.8933       Region 43
...
```

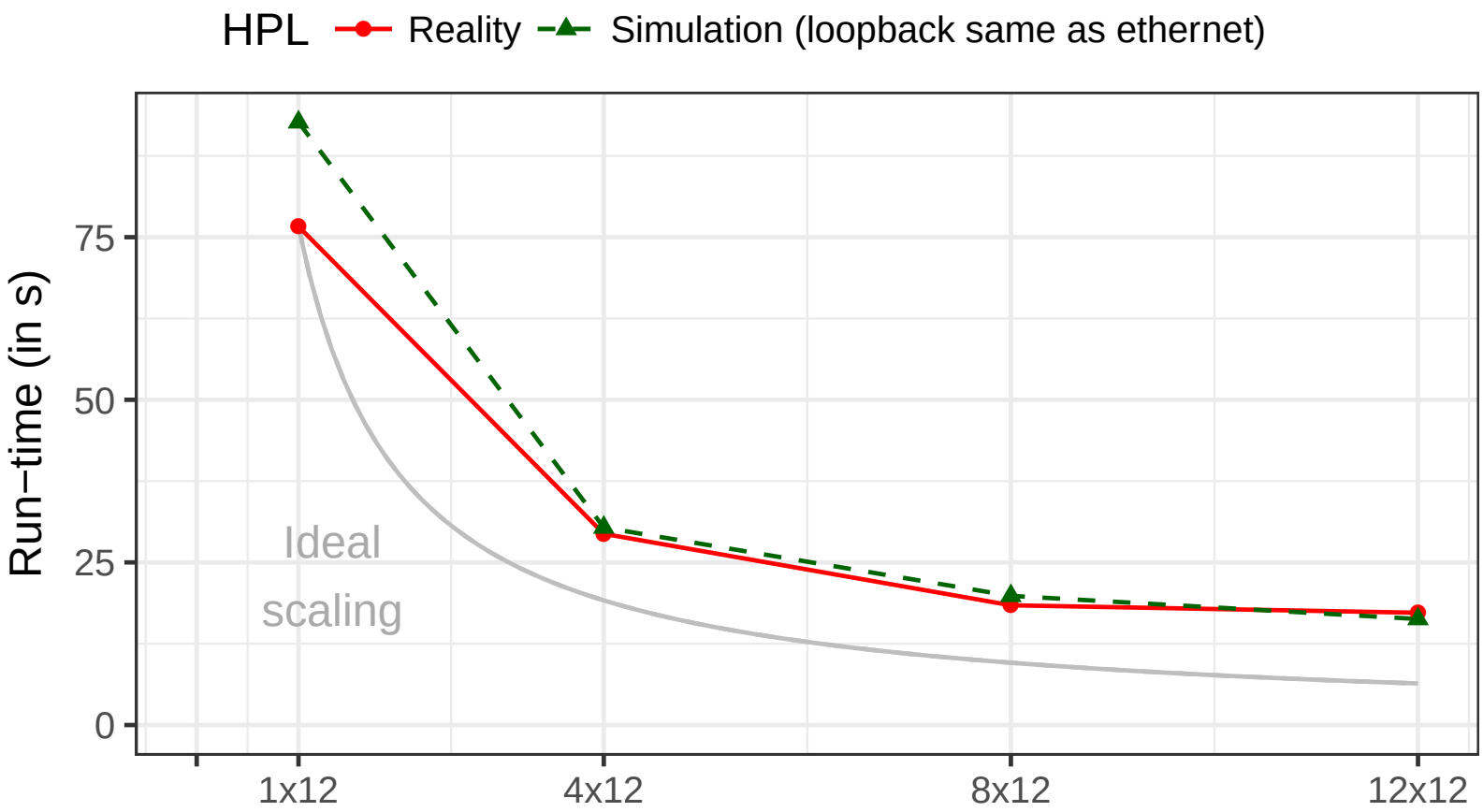
# Contribution 2: Outcome



# Contribution 3: Problem

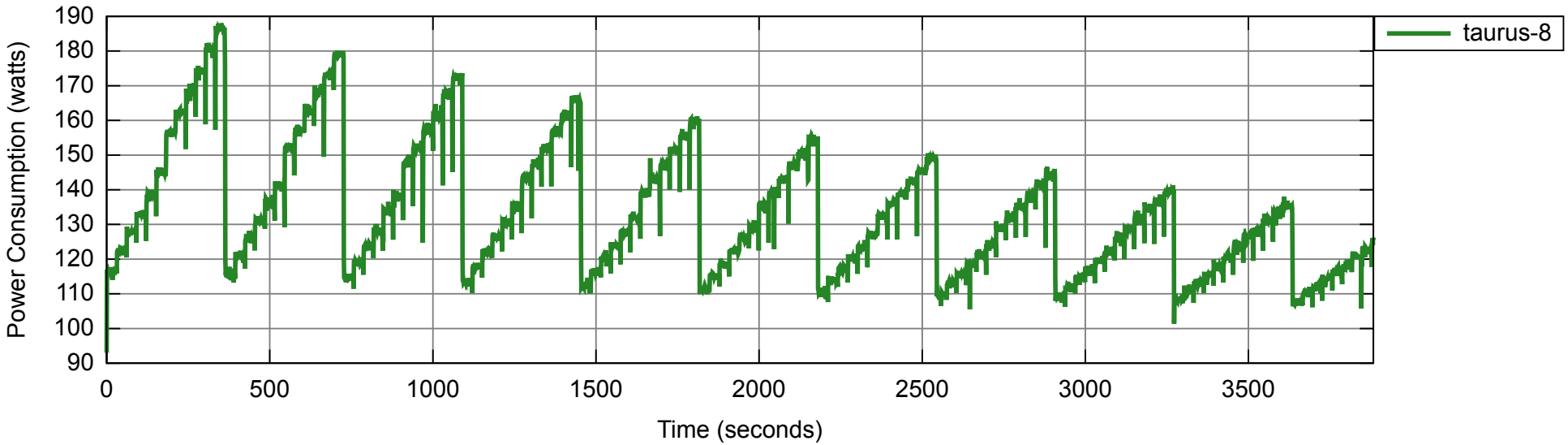
HPL is more complicated than this. Two main issues:

1. HPL sends many large messages from rank to rank.
   $\rightsquigarrow$ faster intra-node communications should be accounted for



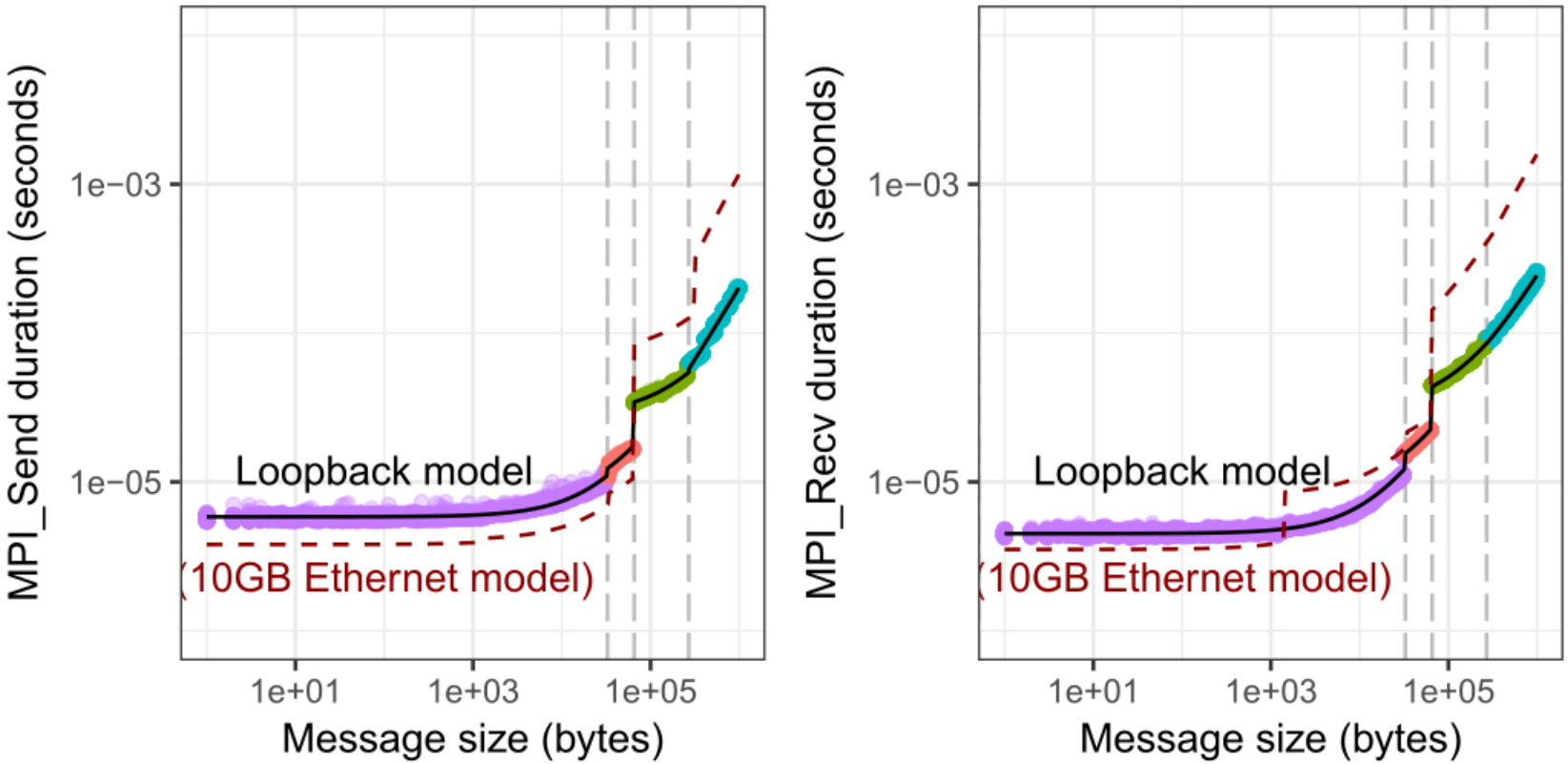
# Contribution 3: Problem (2/2)

2. Makes **heavy** use of `MPI_Iprobe` in order to run computations while waiting for data.
   - But `Iprobes` **do** consume significant amounts of energy!
   - We hence cannot ignore `Iprobes`!



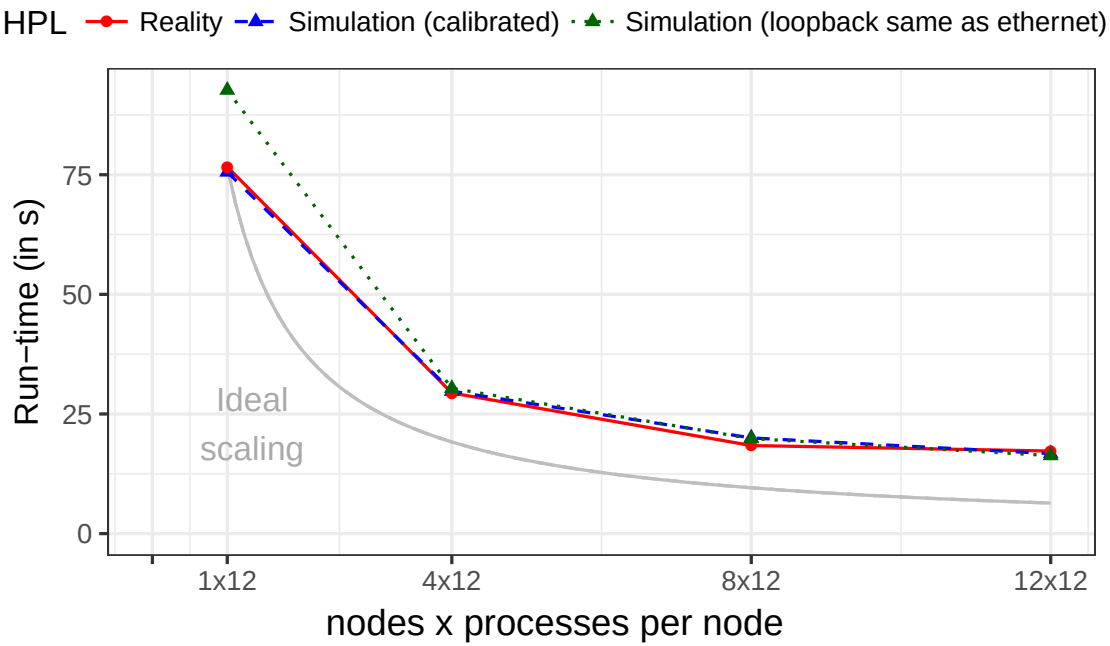
# Contribution 3: Solution

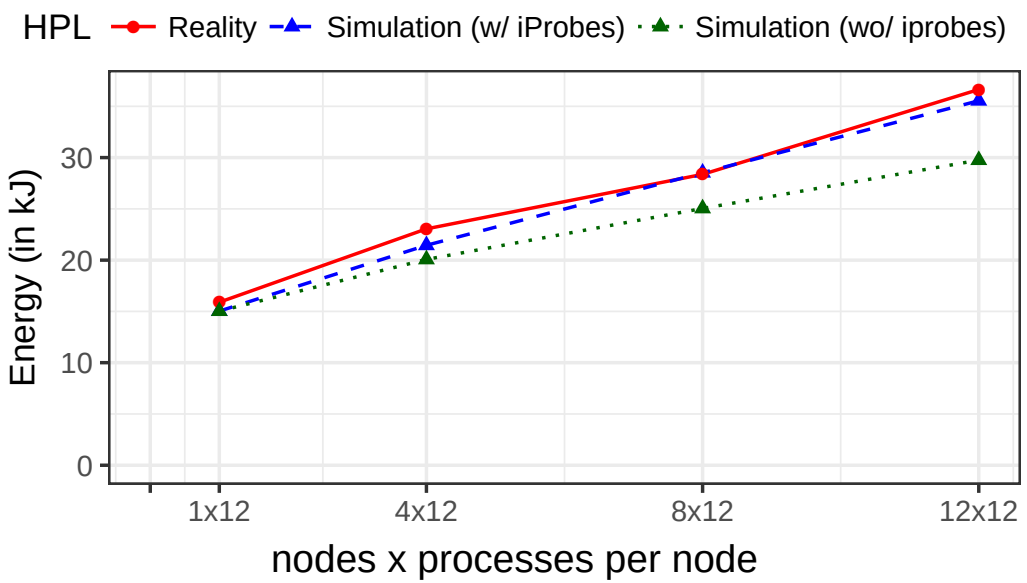
1. Calibrate loopback usage by sending local messages



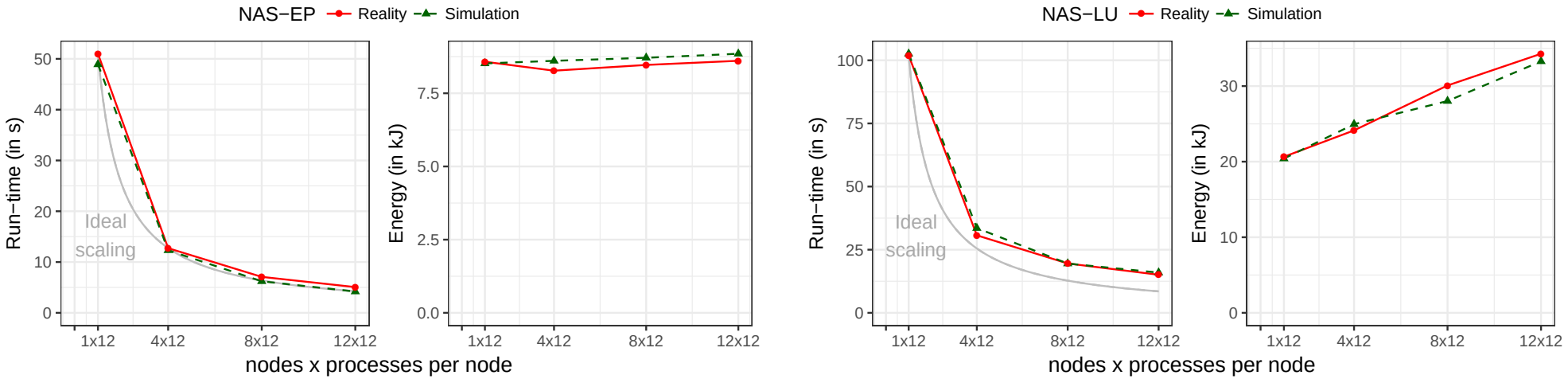

2. `Iprobe` issue is simple: Scale CPU usage while iprobeing via parameter `-cfg=smpi/iprobe-cpu-usage` (here: 0.61)

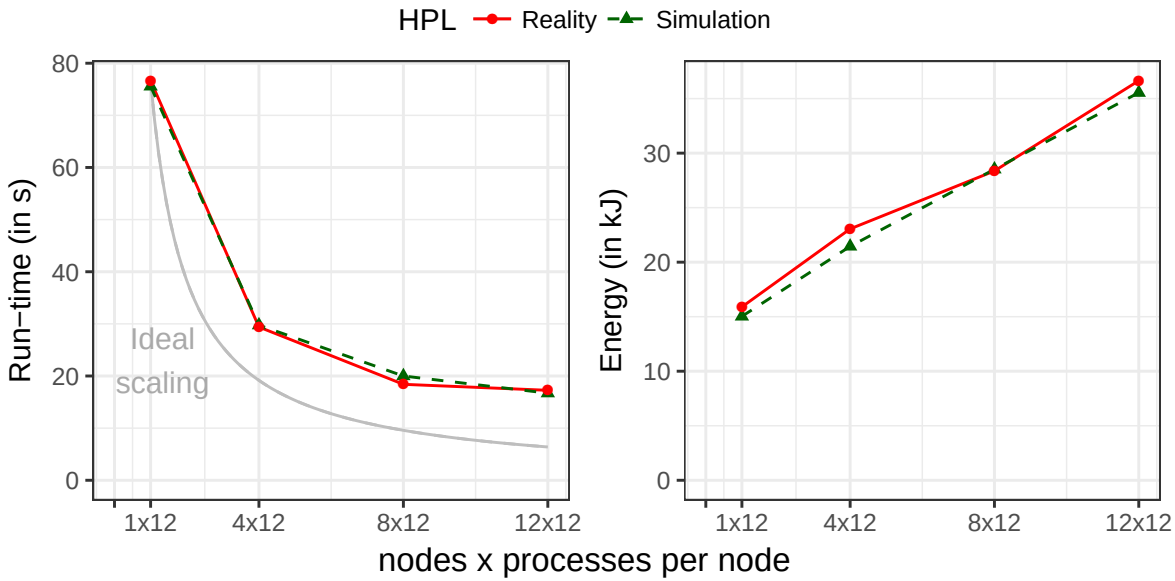
# Contribution 3: Outcome



# Validation Recap



# Take-aways

SimGrid can be helpful to your research

- **Versatile**: Several communities (Scheduling, Grids, HPC, P2P, Clouds)
- **Accurate**: Model limits known thanks to validation studies
- **Sound**: Easy to use, extensible, fast to execute, scalable, well tested
- **Open**: LGPL; User-community much larger than contributors group
- Around since 18 years, ready for at least 18 more years


- **Discover**: http://simgrid.gforge.inria.fr/
- **Learn**: tutorials, user manual and examples
- **Join**: mailing list, #simgrid on irc.debian.org