Energy modeling and optimization for HPC

A. Guermouche, J.-P. Halimi, A. Laurent, A. Mazouz, **B. Pradelle**, N. Triquenaux, W. Jalby

Energy at UVSQ

- As part of the PerfCloud project
  - 6 post-doc, PhD studdent, engineers
  - Formerly at Exascale Computing Research

- Saving energy in HPC since 2011

- Software solutions to save energy

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

How to save energy?

$$e = P_{avg} \times t$$

How to save energy?

$$e = P_{avg} \times t$$

- Reducing the execution time saves energy

- Apply one of the many existing performance optimization techniques

How to save energy?

$$e = P_{avg} \times t$$

- Energy is also saved when saving power
  - ...while maintaining performance !
- We use DVFS to reduce P_{avg}

What is DVFS?

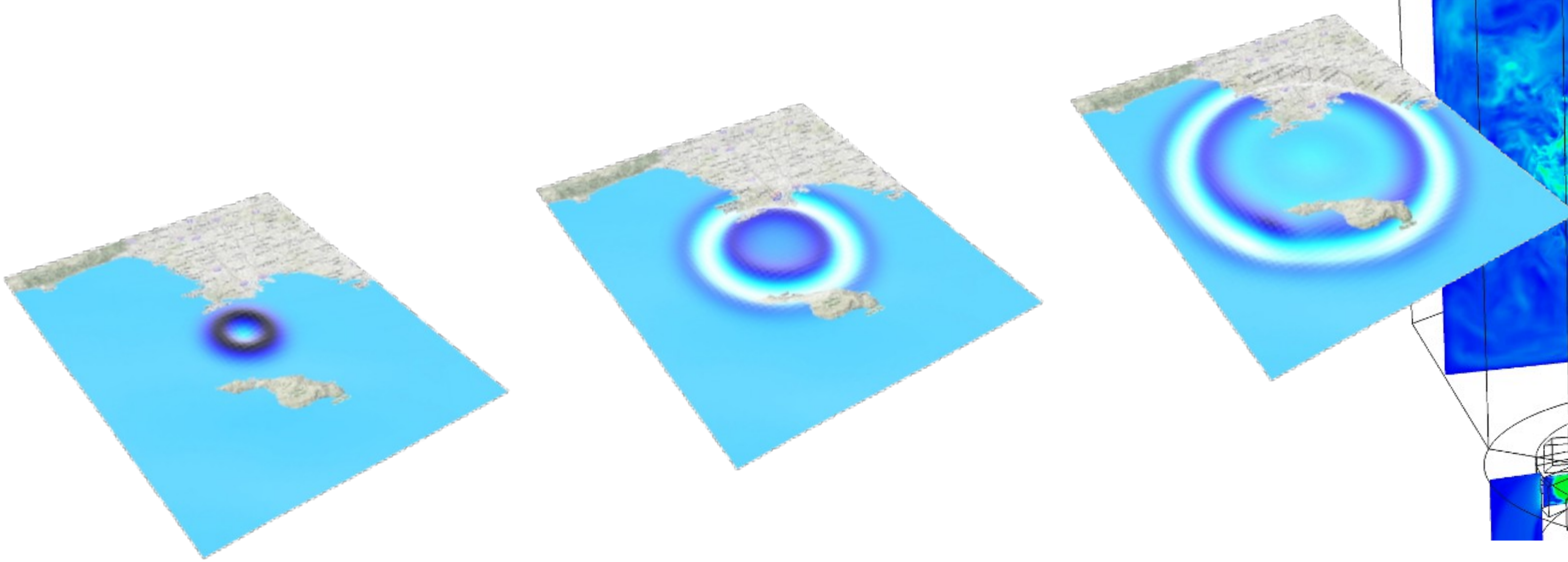
- Dynamic Voltage and Frequency Scaling
- Manually control CPU frequency
  - Also impacts CPU voltage (hardware decides)
  - Low frequency = low power consumption

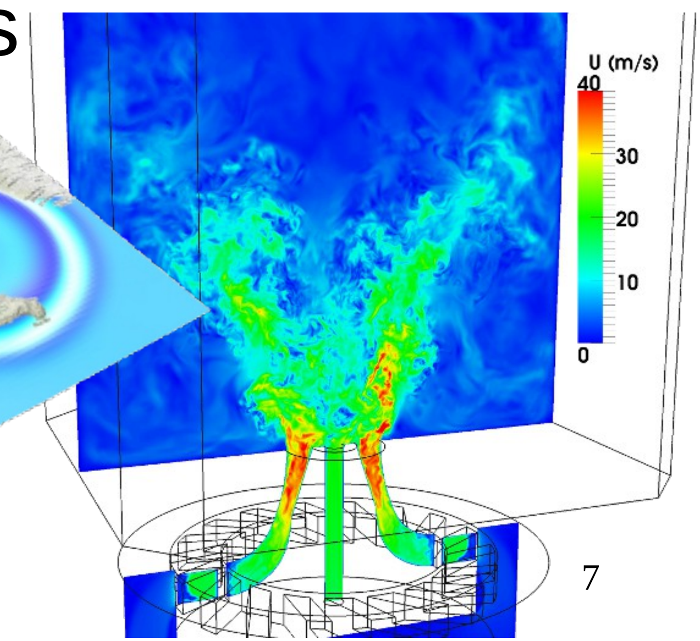
The lowest frequency is not always

the most energy efficient one

Target HPC programs

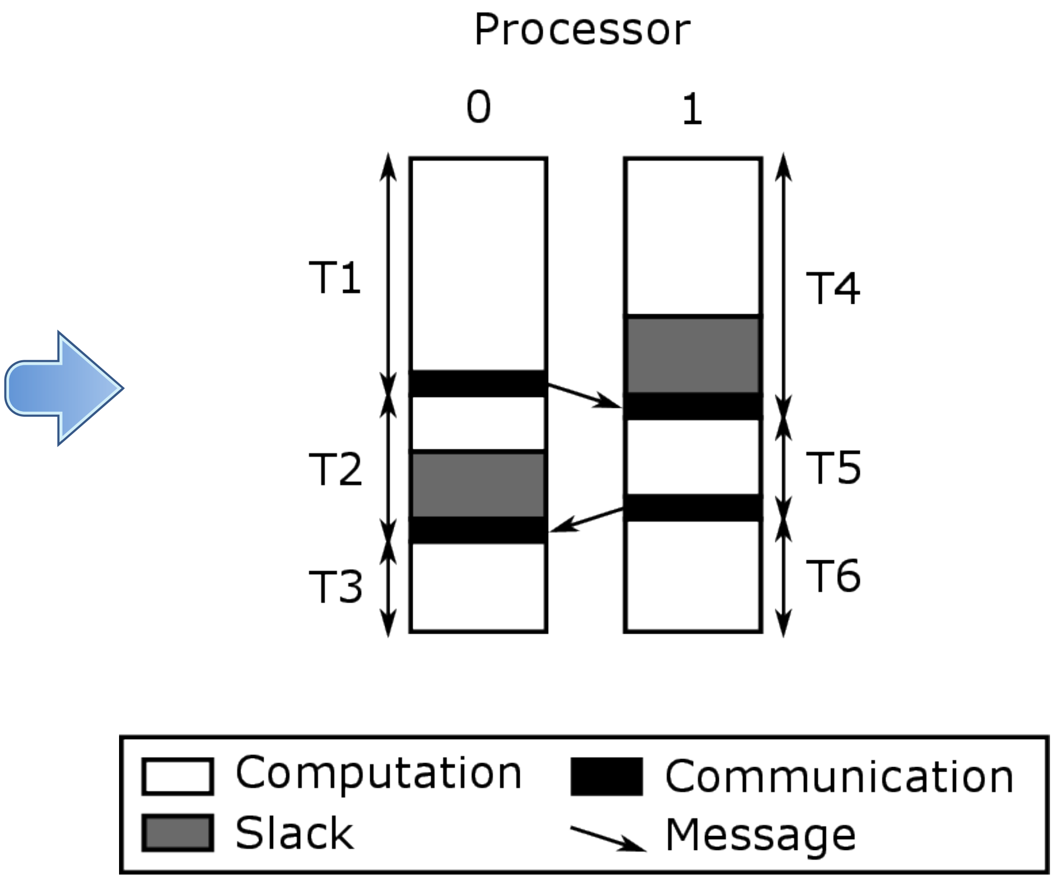
- Use message-passing (MPI) for parallelization
- Focus on mostly-iterative programs
  - A few loops with many iterations
  - Stable communication/computation pattern
- Typical targets: simulation codes



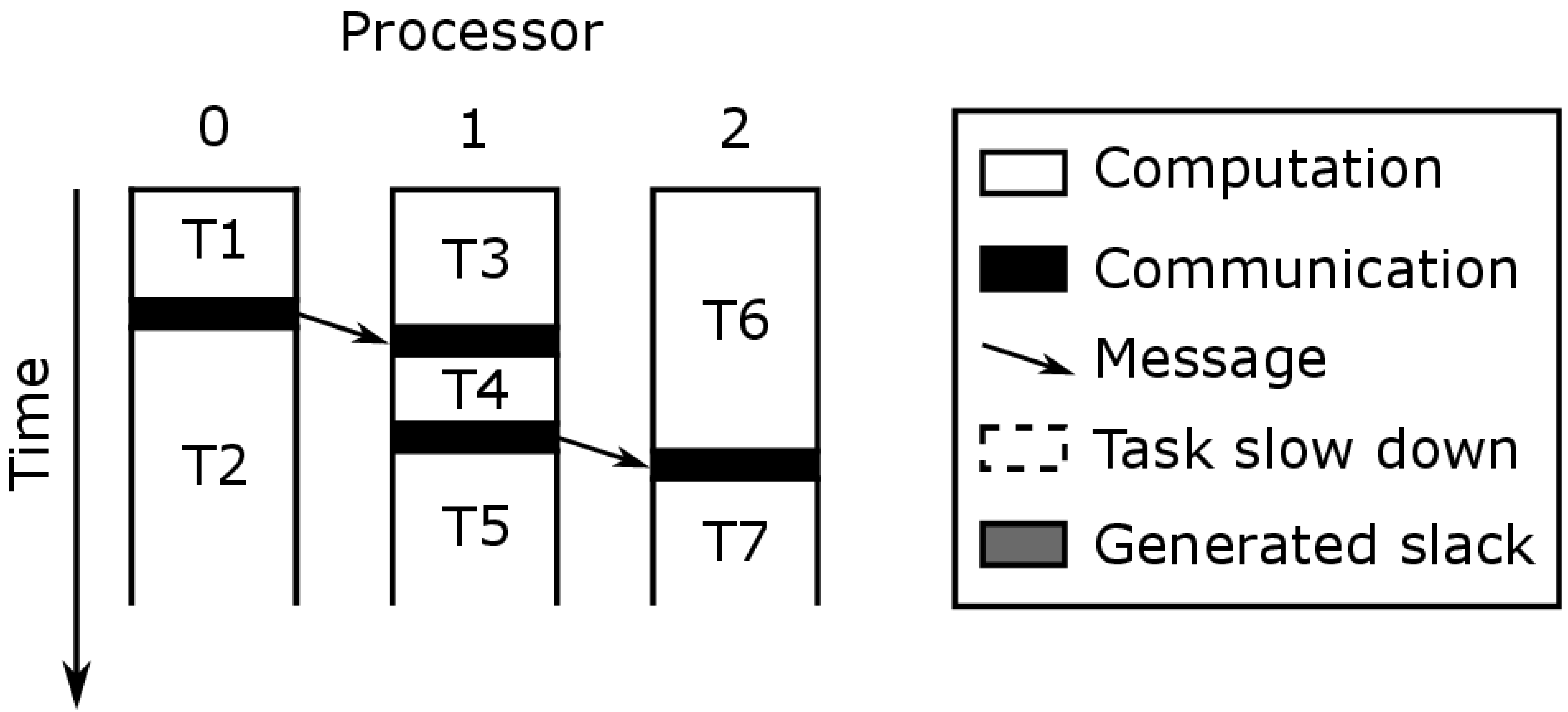


Task graph

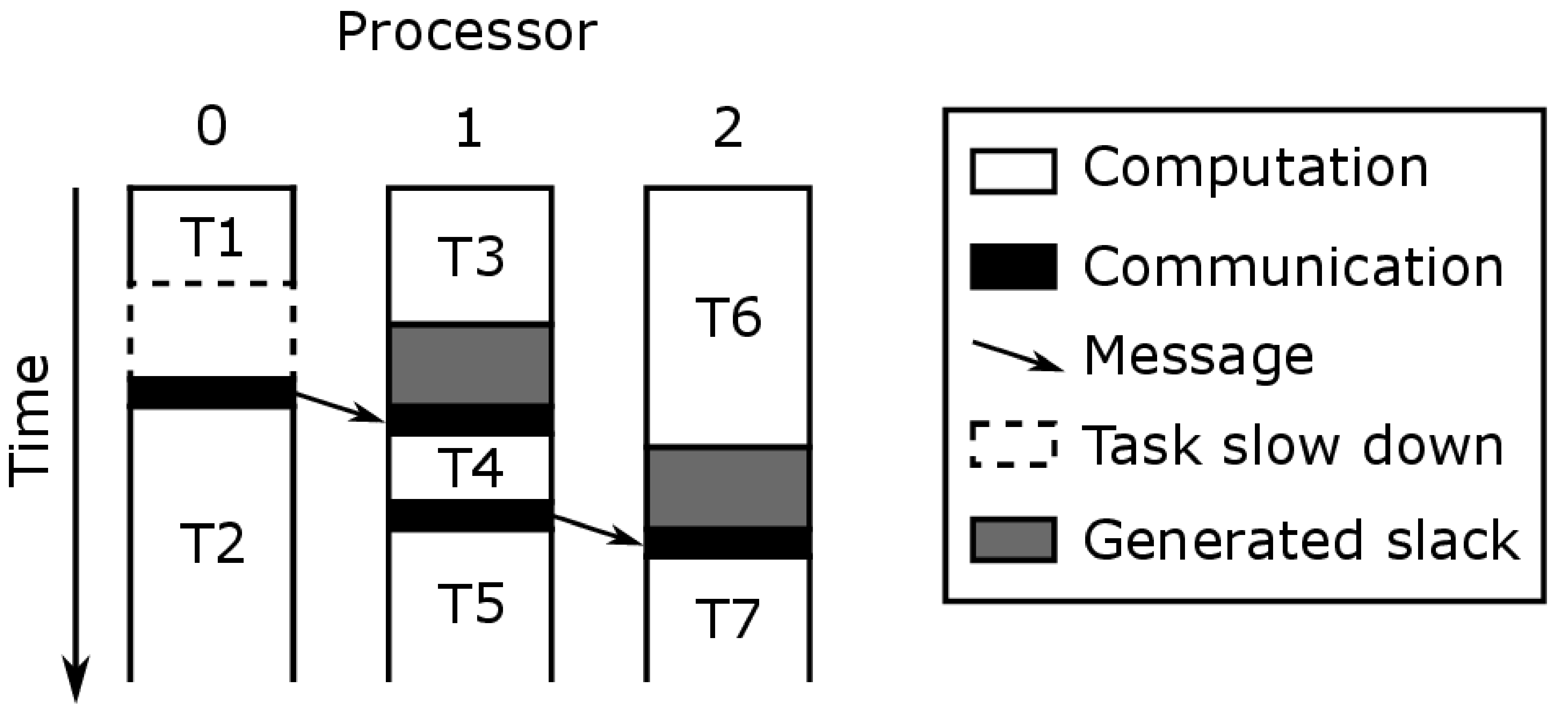
```
for (t = 0; t < T; t++) {
  if (rank == 0) {
    ... (T1)
    MPI_Send(1, ...)
    ... (T2)
    MPI_Recv(1, ...)
    ... (T3)
  } else {
    ... (T4)
    MPI_Recv(0, ...)
    ... (T5)
    MPI_Send(0, ...)
    ... (T6)
  }
}
```



DVFS and tasks

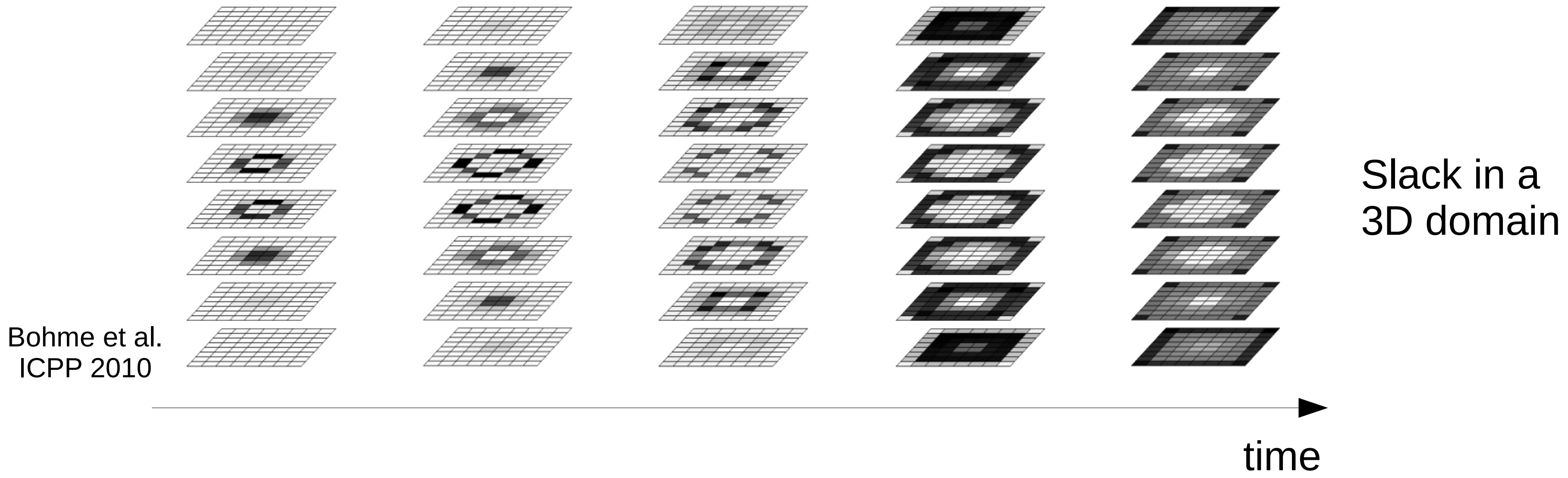


DVFS and tasks



Slack and energy

- A slowdown in a process may propagate to others



- Slack in MPI = active polling
  - Very high power consumption !

Slack and energy

❗ Slack is bad for energy ❗

We must avoid it when performing DVFS

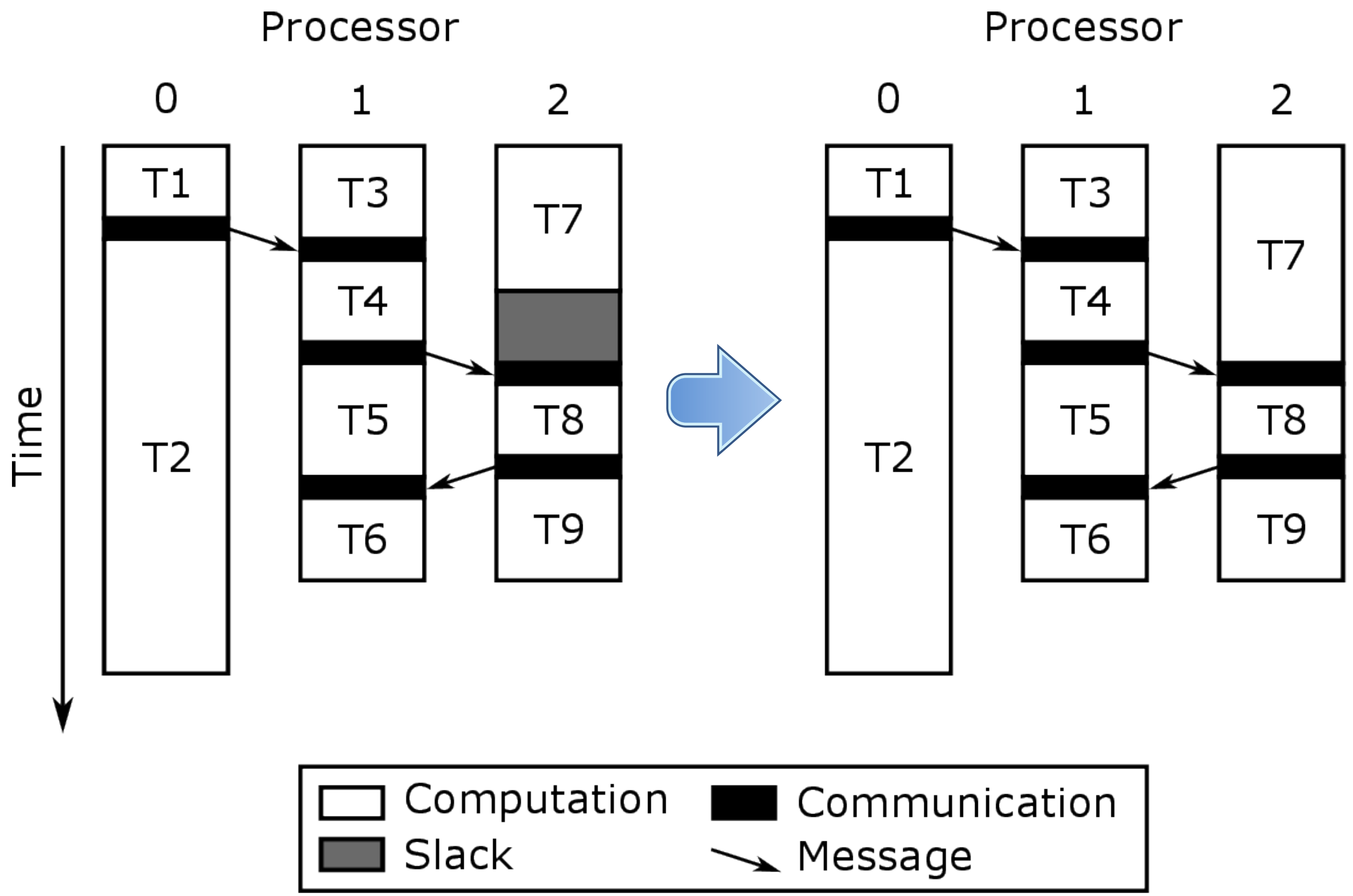
Existing solutions

- Avoid slack in all cases
  - Reduce frequency during slack
  - Slow down tasks out of the critical path

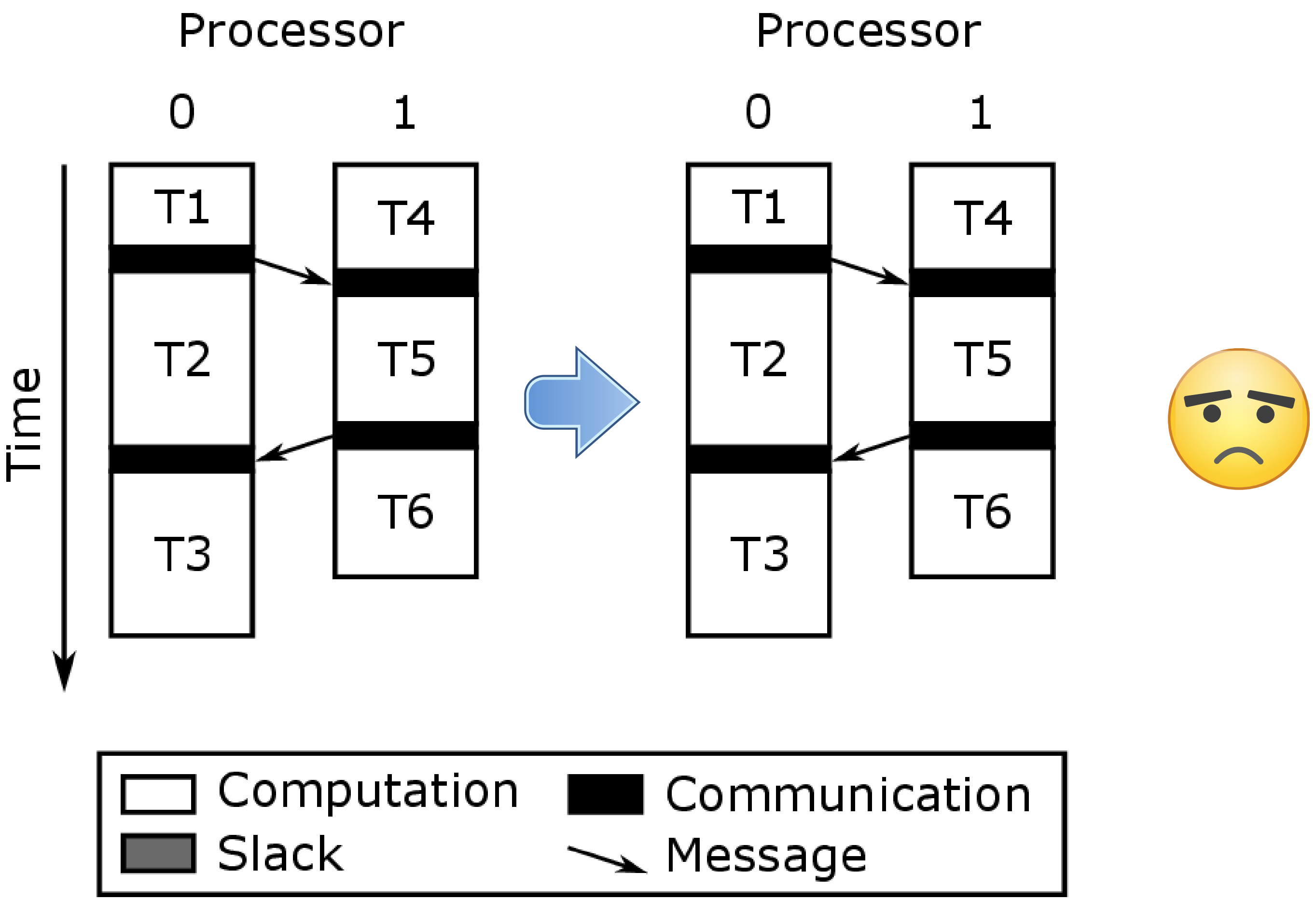
    (= those with slack)

- Slow down whole iterations: Jitter
- Slow down individual tasks: Adagio
  - State of the art

Adagio


Processor
0
1
2
Time
T1
T2
T3
T4
T5
T6
T7
T8
T9
Processor
0
1
2
T1
T2
T3
T4
T5
T6
T7
T8
T9
Computation
Communication
Slack
Message

Adagio



Processor
0
1
T1
T2
T3
T4
T5
T6
Time
Processor
0
1
T1
T2
T3
T4
T5
T6
Computation
Communication
Slack
Message

Balanced codes

What if some tasks still benefit from a lower frequency?

Let's have a look...

Locally optimal frequency

- Every task has a *locally optimal frequency*
  - Minimizes the task energy consumption
  - Ignores the effects on other tasks

- Which frequency is locally optimal? (for a given task)

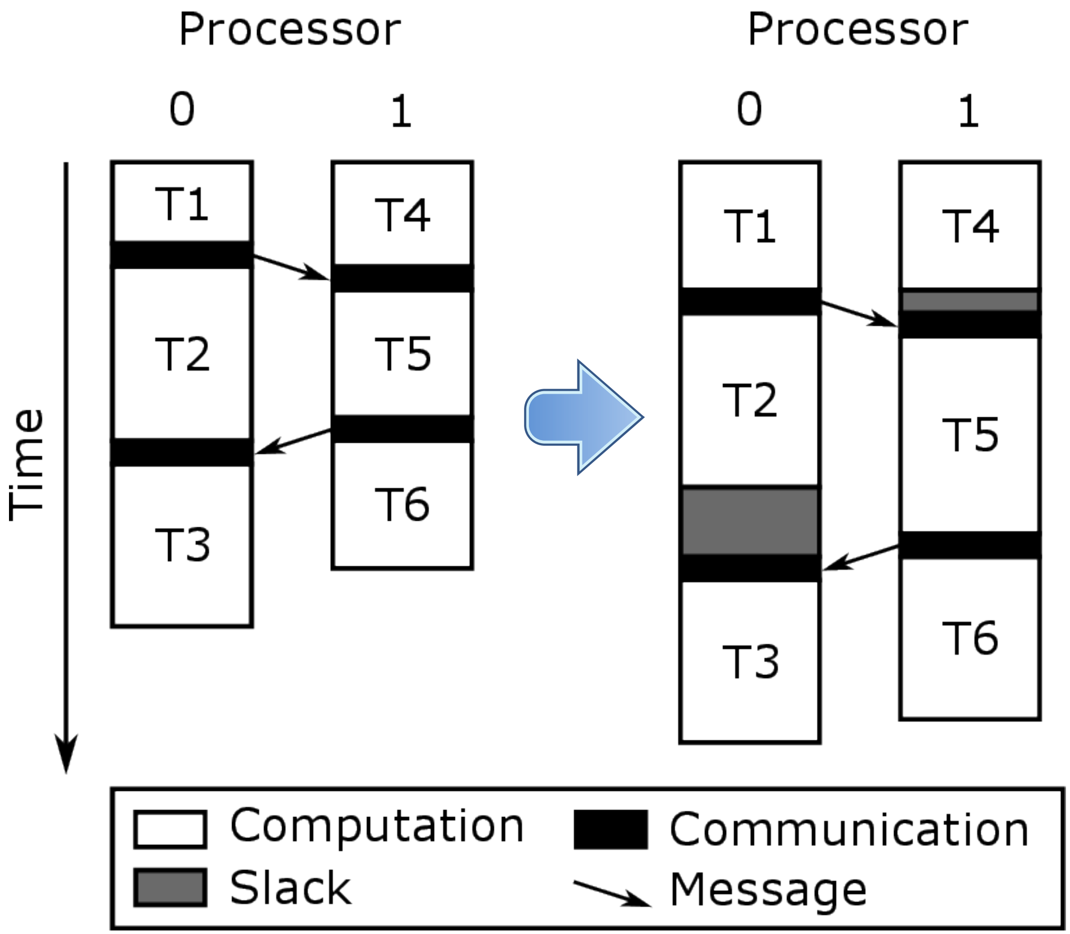
→ how much energy a task consumes

for each frequency?

Predicting e(T,f)

- Remember: $e(T,f) = P_{avg}(T,f) \times t(T,f)$
- Predicting t(T,f)
  - Let several loop iterations run
  - Reduce the frequency before every iteration
  - Measure t(T,f) for every T and f
- Predicting P(T,f)
  - Cannot measure P(T,f)
  - Approximate it from offline measurements

Locally optimal frequency



Consequences

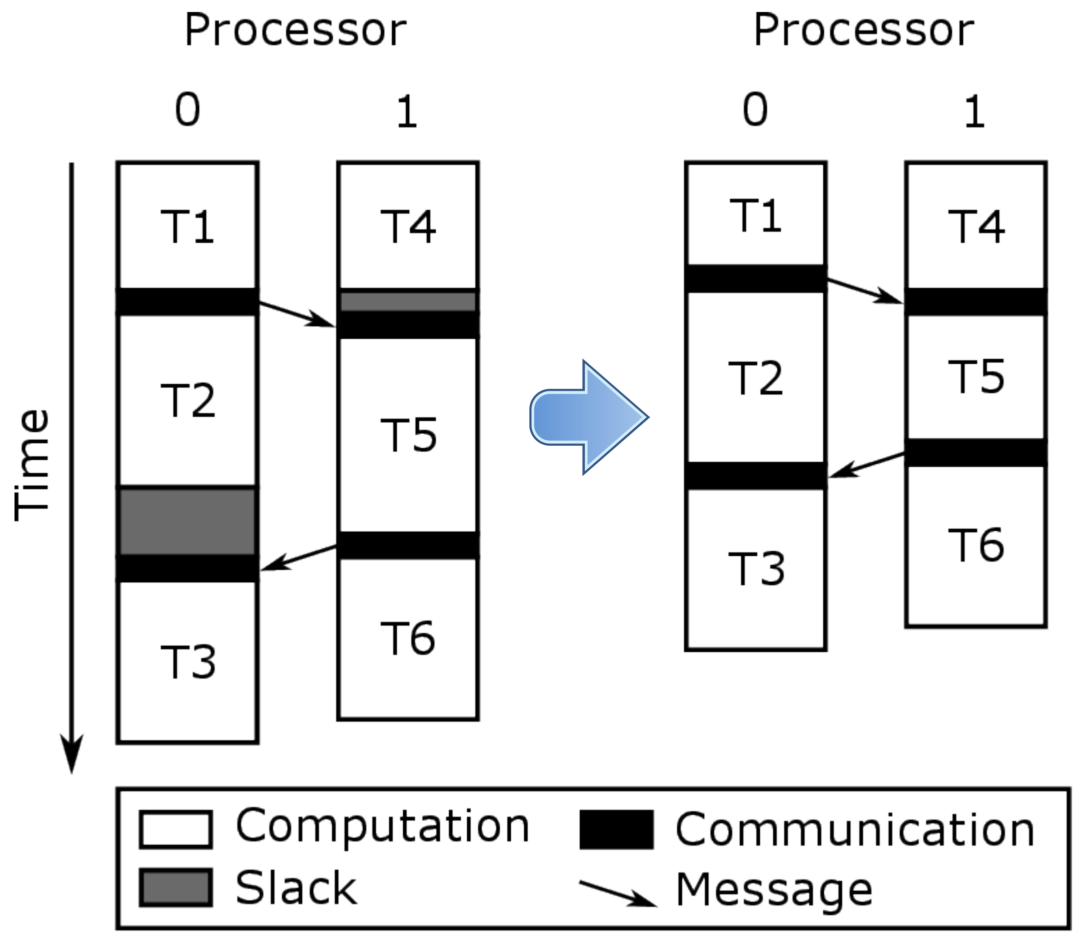
- Some slack may be introduced
- More energy wasted in slack than saved?
  - Complex to evaluate but avoid it in general
- Slow down the task preceding the slack?
- Speed up the task emitting the message?

Globally optimal frequency

- Processes request speedup to others
  - Separate MPI communicator
  - Asynchronous messages
  - Only a few messages exchanged

- Then applied for the rest of the loop execution

Globally optimal frequency

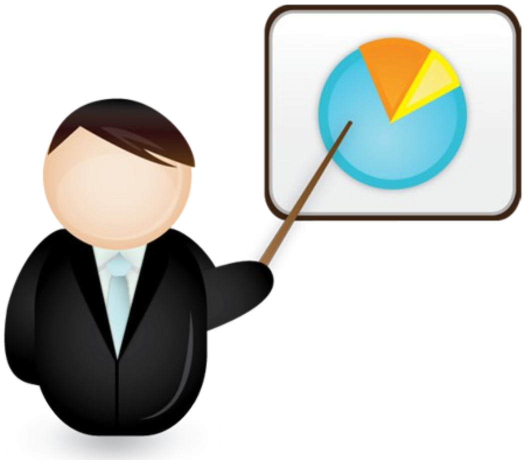


FoREST-mn in short

- Offline profiling
- First iterations while measuring execution time
  - Frequency decreased
- Compute locally optimal frequencies
- Apply them for one iteration
- Converge toward globally optimal frequencies
- Apply the frequency schedule

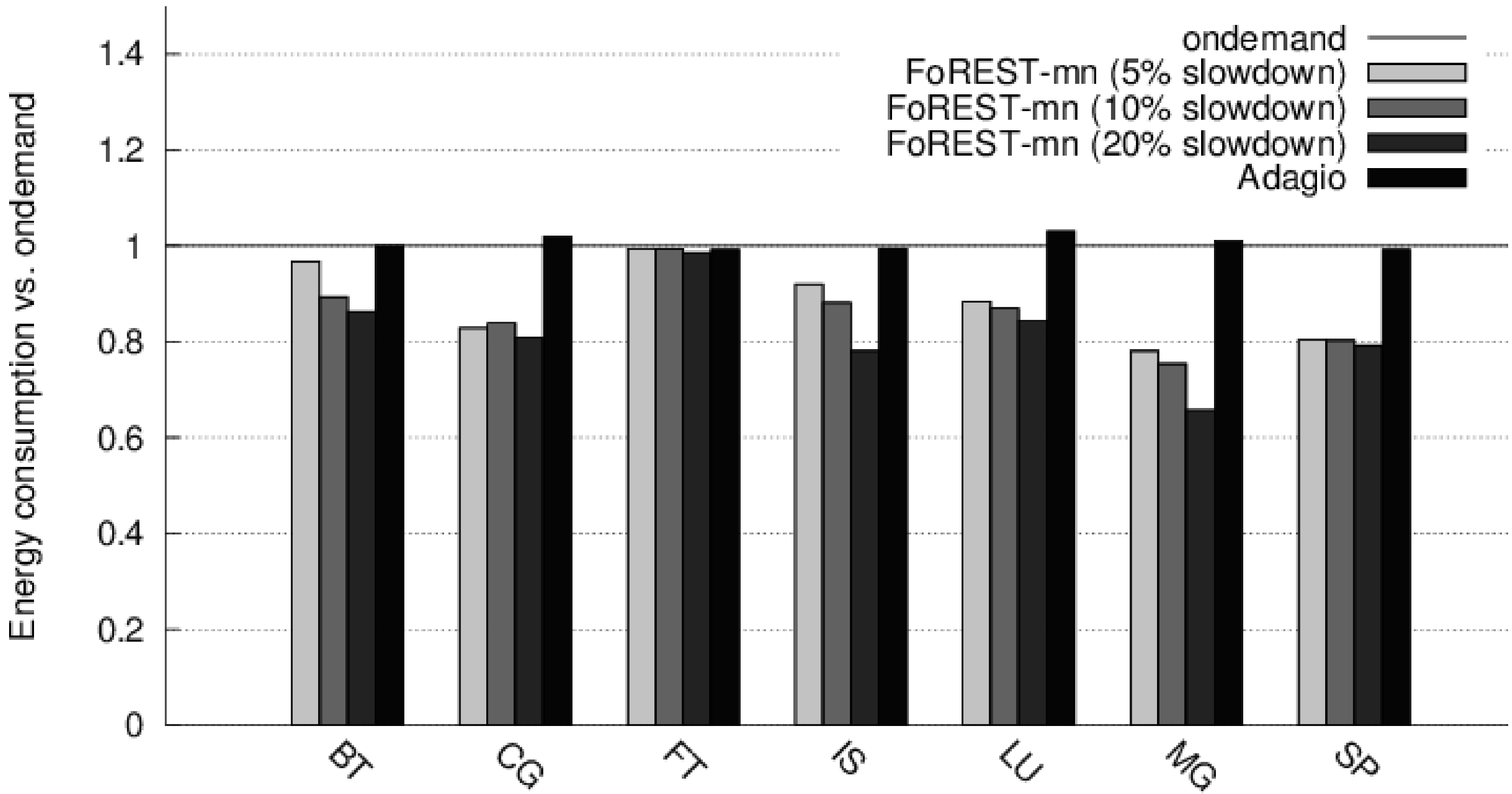
Experiments

- 4 servers (Strasbourg)
  - 2x8 cores Intel SandyBridge
  - 64 processor cores

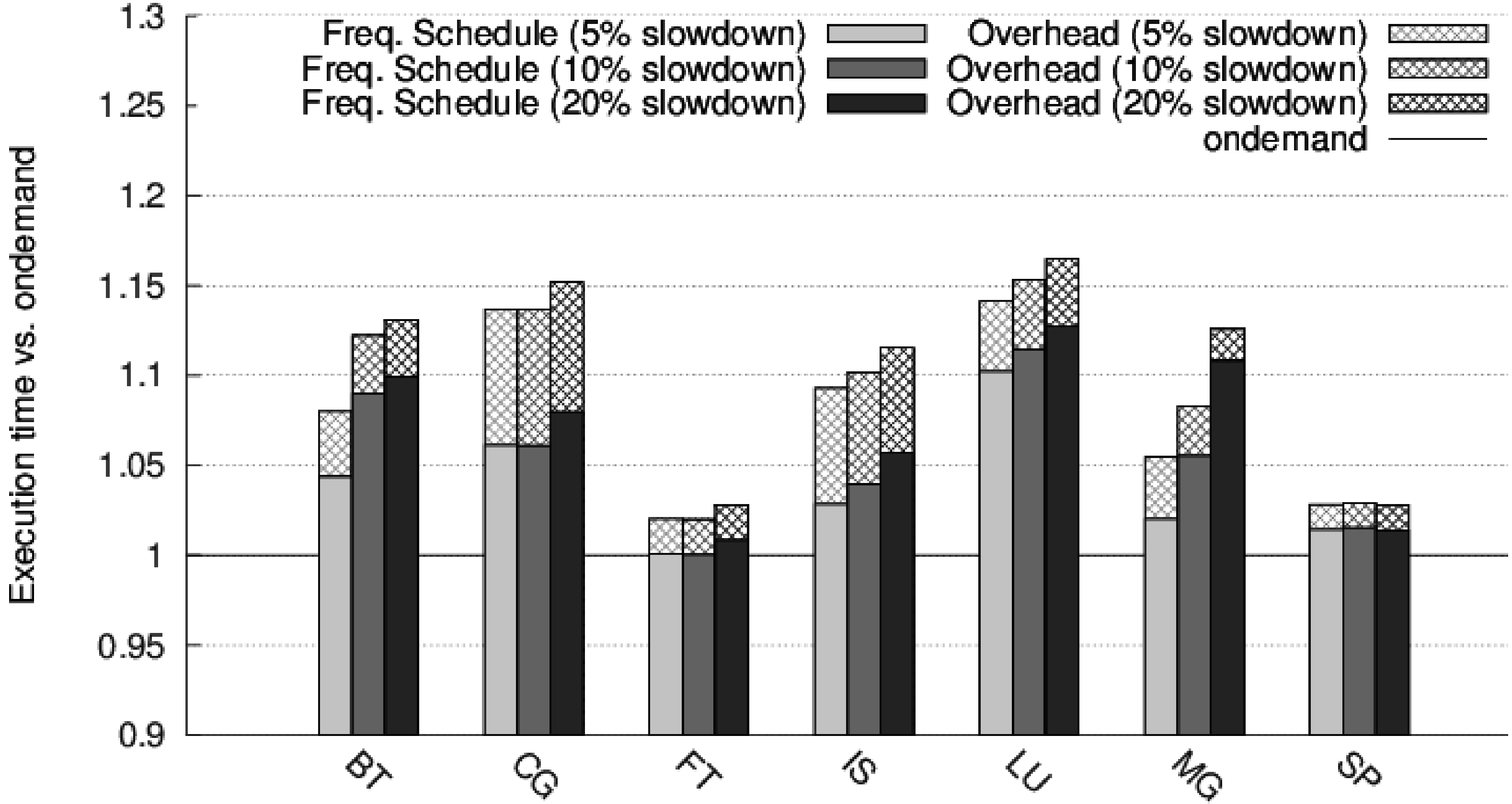


- NAS MPI 3.3.1
  - D class
  - EP excluded
- CPU energy
  - From Intel RAPL

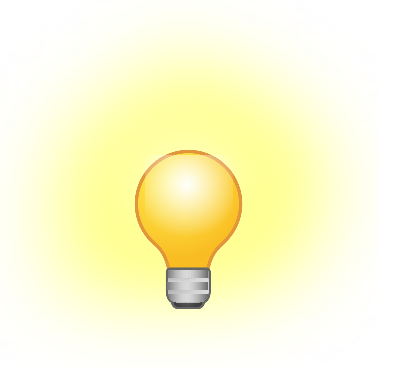
CPU energy consumption



Execution time



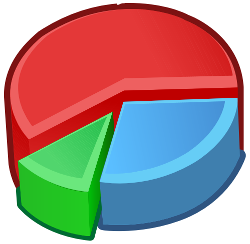
Can we improve it?



- Predict e(T,f) more precisely
  - Use energy modeling (WIP)
- Reduces overhead
- Prediction from tasks characteristics
  - Hardware counters

Current energy model

- Multiple linear regressions
- IPC
  - Accounts for most computations
- Memory traffic (RAM, L3, L2, L1)
- Regression from synthetic benchmarks
  - Various data sizes
  - Various number of active cores
  - Various frequencies



Current energy model

- Good prediction for simple loops (NR)
  - Evolves to support more complex programs
  - Current average error: 3%

- Ultimate goal: accuracy for complex workloads
  - In complex environment (multicore processors)
  - Integration into FoREST-mn

How good is FoREST-mn?

- How much energy can I save?
  - For my HPC program

- OutReach computes it
  - Based on execution traces
  - Maximal energy saving with DVFS
  - Ideal frequency sequence

OutReach

- Gather performance and energy traces
  - For every frequency
- Build the task graph from traces
- Express the optimization problem using LP
  - Solve it
  - Enhance it
  - Solve it
  - ...

OutReach

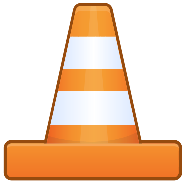
- Gather performance and energy traces
  - For every frequency
- Build the task graph from traces
- Express the optimization problem using LP
  - Solve it
  - Enhance it
  - Solve it
  - ...

Work in progress

Conclusion

- FoREST-mn
  - Significant energy savings
  - Configurable tolerated slowdown
  - Multicore processors support

- Energy modeling effort in progress



- OutReach for complete evaluation

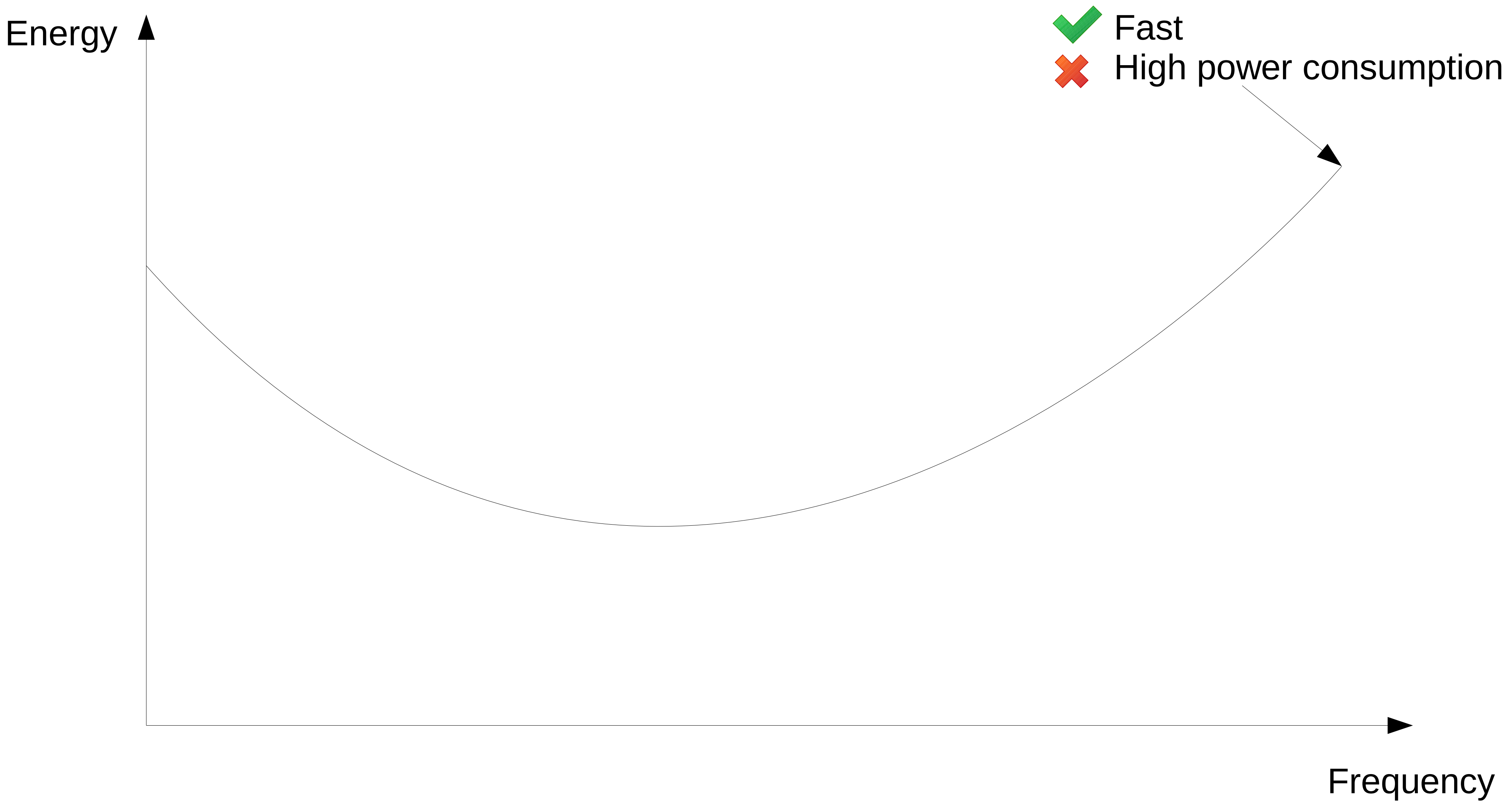
Predicting P(T,f)

- Remember: $P \approx P_{static} + \frac{1}{2} \times A \times C \times V^2 \times f$
- Assume: $P_{static} \approx k \times (\frac{1}{2} \times A \times C \times V^2 \times f)$
- Thus: $P \approx (k+1) \times (\frac{1}{2} \times A \times C \times V^2 \times f)$

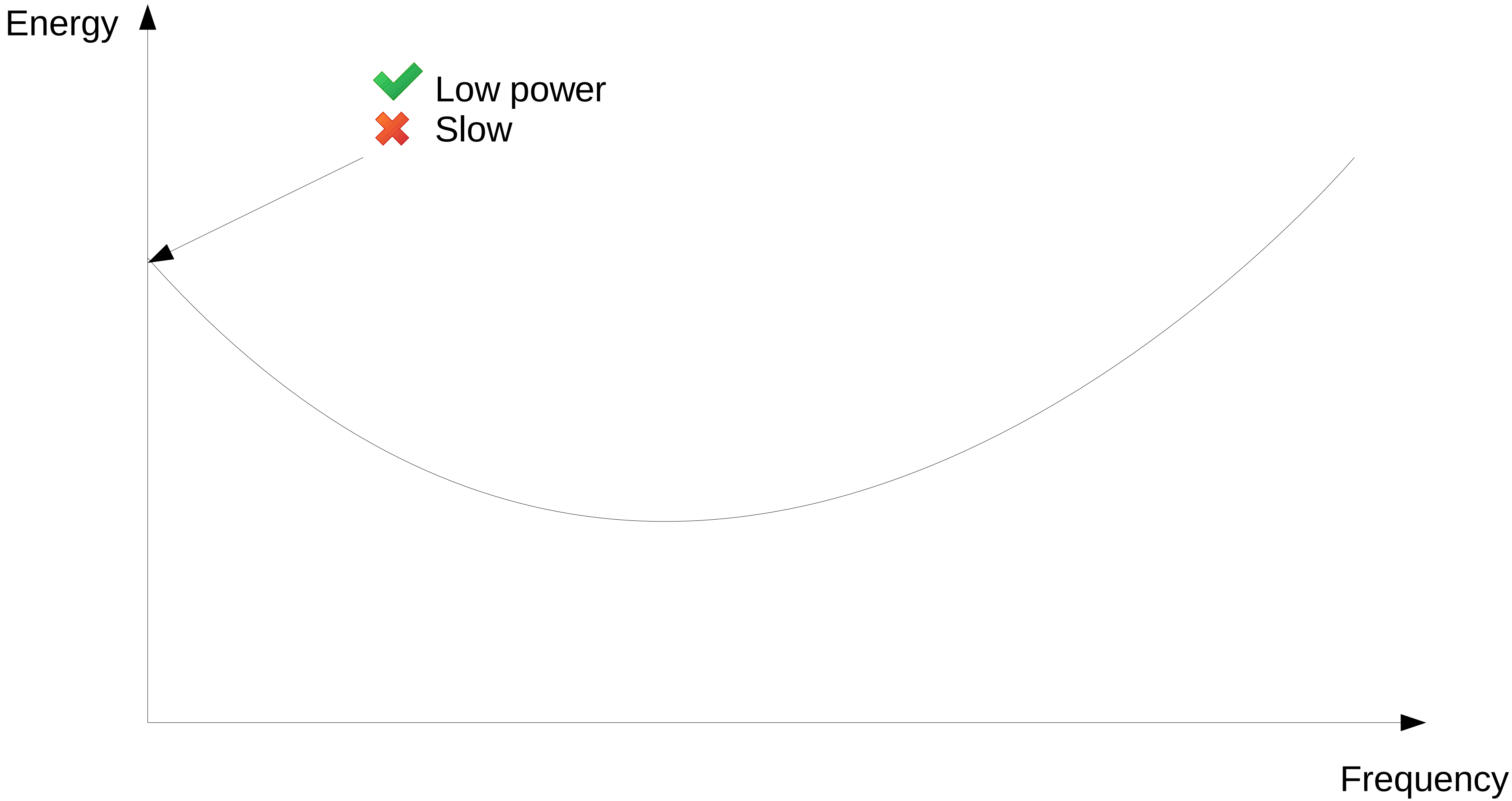
$$\frac{P(f_1)}{P(f_2)} \approx \frac{(k_1+1) \times (\frac{1}{2} \times A \times C_1 \times V_1^2 \times f_1)}{(k_2+1) \times (\frac{1}{2} \times A \times C_2 \times V_2^2 \times f_2)} = \frac{(k_1+1) \times (\frac{1}{2} \times C_1 \times V_1^2 \times f_1)}{(k_2+1) \times (\frac{1}{2} \times C_2 \times V_2^2 \times f_2)}$$

Only architectural parameters remain

Typical energy profile



Typical energy profile



Typical energy profile