Minimizing Energy Consumption of MPI Programs in Realistic Environment

Nicolas Triquenaux[1], Amina Guermouche[1], Benoît Pradelle[1],
Jean Christophe Beyler[2],William Jalby[1]

1 Université de Versailles Saint-Quentin-en-Yvelines, Versailles, France
2 Intel Corporation

July 1^{st}, 2014

Introduction

- Computational power

  Tianhe-2 $33,863$ TFlops

- Increase in the number of components

  Tianhe-2 $3,120,000$ cores

- Large power consumption on powerful supercomputers

  Tianhe-2 $17MW$

  (Chad power consumption : $\sim 10MW$)

Introduction

$$Energy = Power \times Execution\_Time$$
$$Power = f(frequency^3)$$

- Dynamic Voltage and Frequency scaling (DVFS)
  - Reduce frequency :

  → Reduce power consumption

  → Reduce energy consumption

  if the slowdown due to lower frequency does not dramatically increase execution time

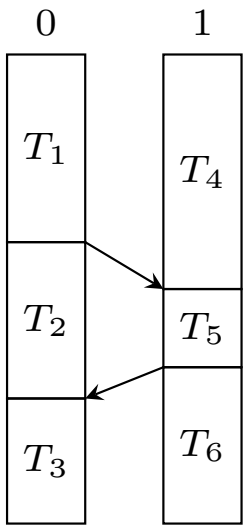
Outline

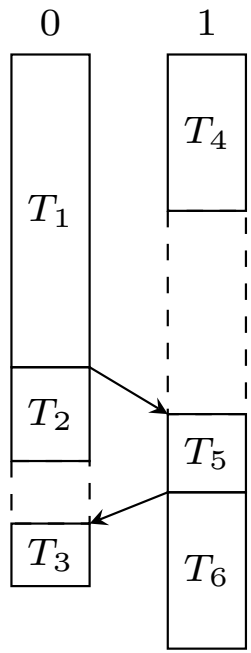
Model

- MPI Applications
  - Task graph
    (task : computations between two communications)
- Dynamic Voltage and Frequency Scaling (DVFS)
  - Discrete set of frequencies
  - Frequency transition overhead
- Multi-node architecture
  - Same frequency on cores within the same processor

Frequency impact on execution time

- Slowing down one task may increase the whole execution time

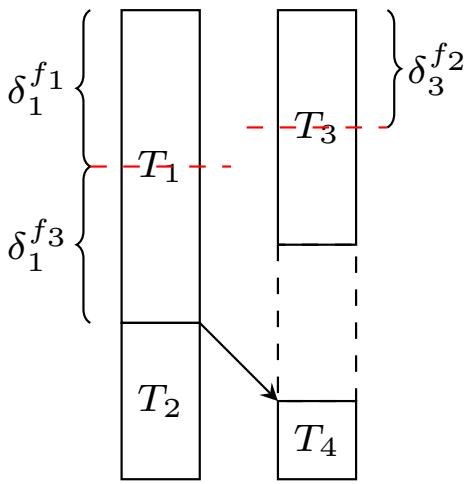




Offline Scheduling : existing linear programming solution

[Rountree & *al*, SC'07]

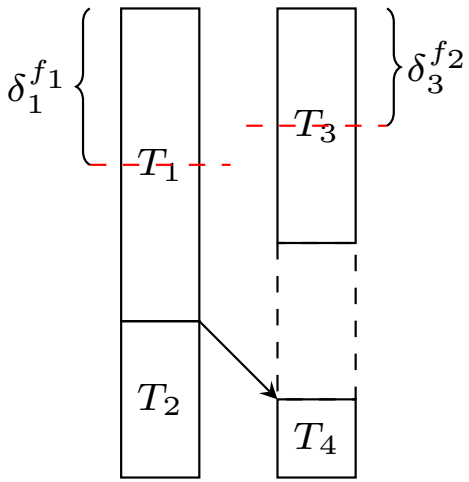
- Precedence constraints
- P_i^f and $Execution\_Time_i^f$ of task T_i at frequency f are known
- $E = \sum_i P_i \times Execution\_Time_i$
- A task can be divided into several portions, each one can be executed at a specific frequency
  - δ_i^f : fraction of time task T_i spends at frequency f
  - $Execution\_Time\_i = \sum_i \delta_i^f \times Execution\_Time_i^f$



Offline scheduling : Realistic environment

Architecture constraints are not considered :

- Cores within the same processor share the same frequency
  - → Parallel tasks one in processor must be executed at the same frequency
- Frequency transition overhead cannot be ignored
  - → A task may be over before its execution frequency is set

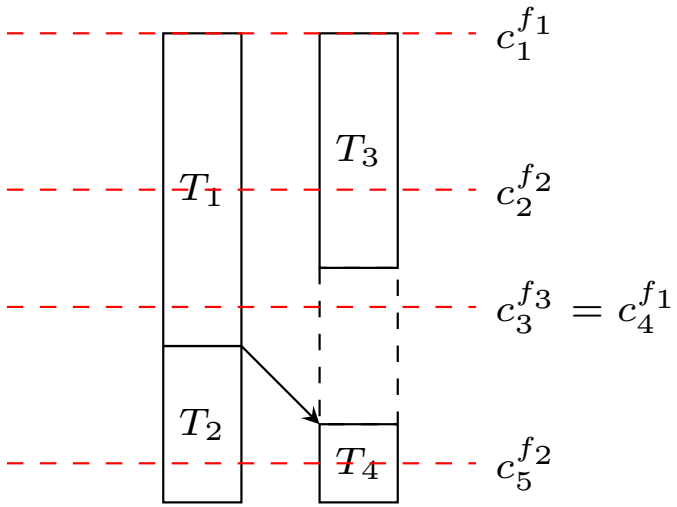


Outline

Linear programming constraints

- $E = \sum_i P_i \times Execution\_Time_i$
- The time a frequency is set ($c_i^{f_j}$)

1. Frequency transition overhead :
   - The time between two frequency transitions is larger than a threshold :
     If f_k is set then :
     $c_{i+1}^{f_j} - c_i^{f_k} \geq threshold$
2. Same frequency over the cores
   - The frequency switch is considered at the processor level, not the task level

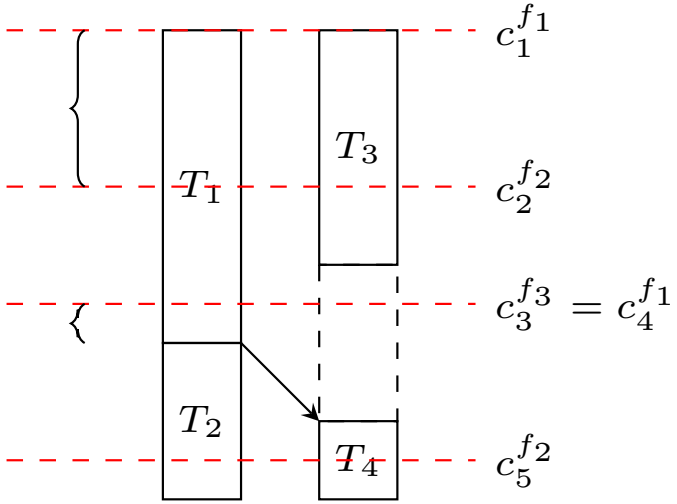


Translating architecture constraints

- sT_i : start time of T_i
- eT_i : end time of T_i

$$exec_1^{f_1} = \begin{cases} c_2^{f_2} - sT_1 & for\ c_1^{f_1} \\ eT_1 - c_4^{f_1} & for\ c_4^{f_1} \end{cases}$$

$$exec_1^{f_2} = \begin{cases} c_3^{f_3} - c_2^{f_2} & for\ c_2^{f_2} \\ 0 & for\ c_5^{f_2} \end{cases}$$

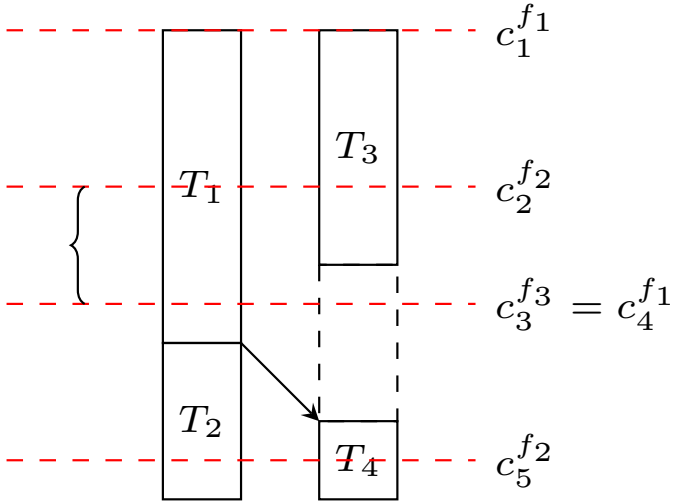


Translating architecture constraints

- sT_i : start time of T_i
- eT_i : end time of T_i

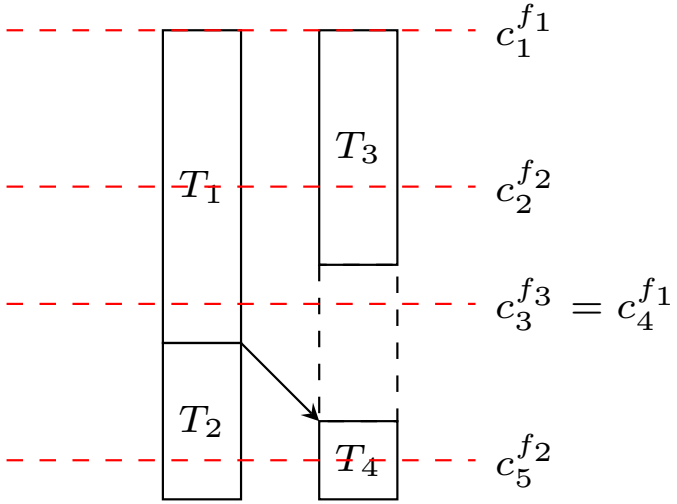
$$exec_1^{f_1} = \begin{cases} c_2^{f_2} - sT_1 & for\ c_1^{f_1} \\ eT_1 - c_4^{f_1} & for\ c_4^{f_1} \end{cases}$$

$$exec_1^{f_2} = \begin{cases} c_3^{f_3} - c_2^{f_2} & for\ c_2^{f_2} \\ 0 & for\ c_5^{f_2} \end{cases}$$



Translating architecture constraints

- sT_i : start time of T_i
- eT_i : end time of T_i



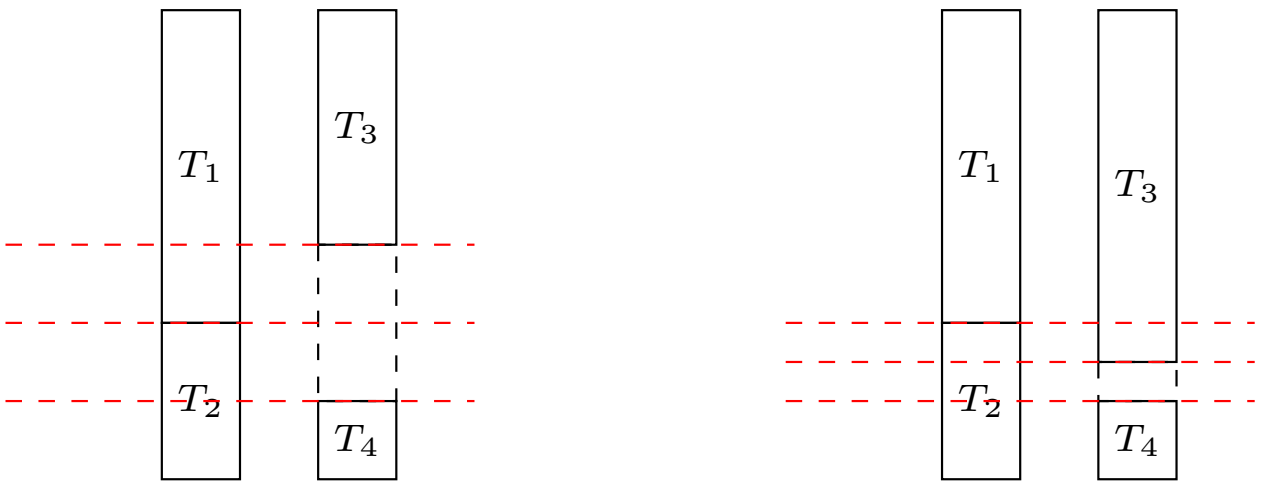
$$exec_1^{f_1} = \begin{cases} c_2^{f_2} - sT_1 & for\ c_1^{f_1} \\ eT_1 - c_4^{f_1} & for\ c_4^{f_1} \end{cases}$$

$$exec_1^{f_2} = \begin{cases} c_3^{f_3} - c_2^{f_2} & for\ c_2^{f_2} \\ 0 & for\ c_5^{f_2} \end{cases}$$

$$exec_i^{f_j} = \begin{cases} min(eT_i, c_{k+1}^{f_{j+1}}) - max(sT_i, c_k^{f_j}) & if \begin{cases} c_k^{f_j} \leq sT_i \leq c_{k+1}^{f_{j+1}} \\ c_k^{f_j} \leq eT_i \leq c_{k+1}^{f_{j+1}} \\ sT_i \leq c_k^{f_j}\ and\ eT_i \geq c_{k+1}^{f_{j+1}} \end{cases} \\ 0 \quad otherwise \end{cases}$$

Another formulation (unusable as well ☹)

- Generate all possible parallel tasks (at different frequencies)
- Find the combination that provides best energy consumption
- ☹ Several giga bytes of possibilites

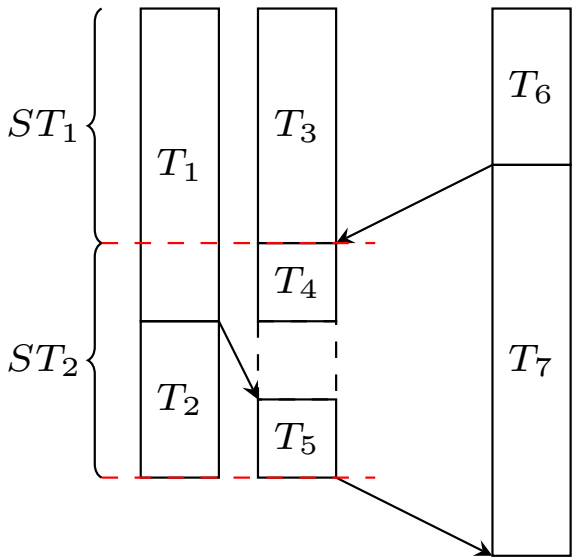


Outline

Principles

- Super tasks (processor level tasks) :
  - Each message sent to or received from a process outside the processor creates a new super task
  - $P_{st} = \sum_{T_i \in st} P_i$
  - $Execution\_time_{st}$: the execution time of the task communicating with the other processor

1. Solve the problem for super tasks using a linear program
2. Apply the solution and redefine the super tasks if necessary
3. Solve the problem with the new task configuration



Energy gain example

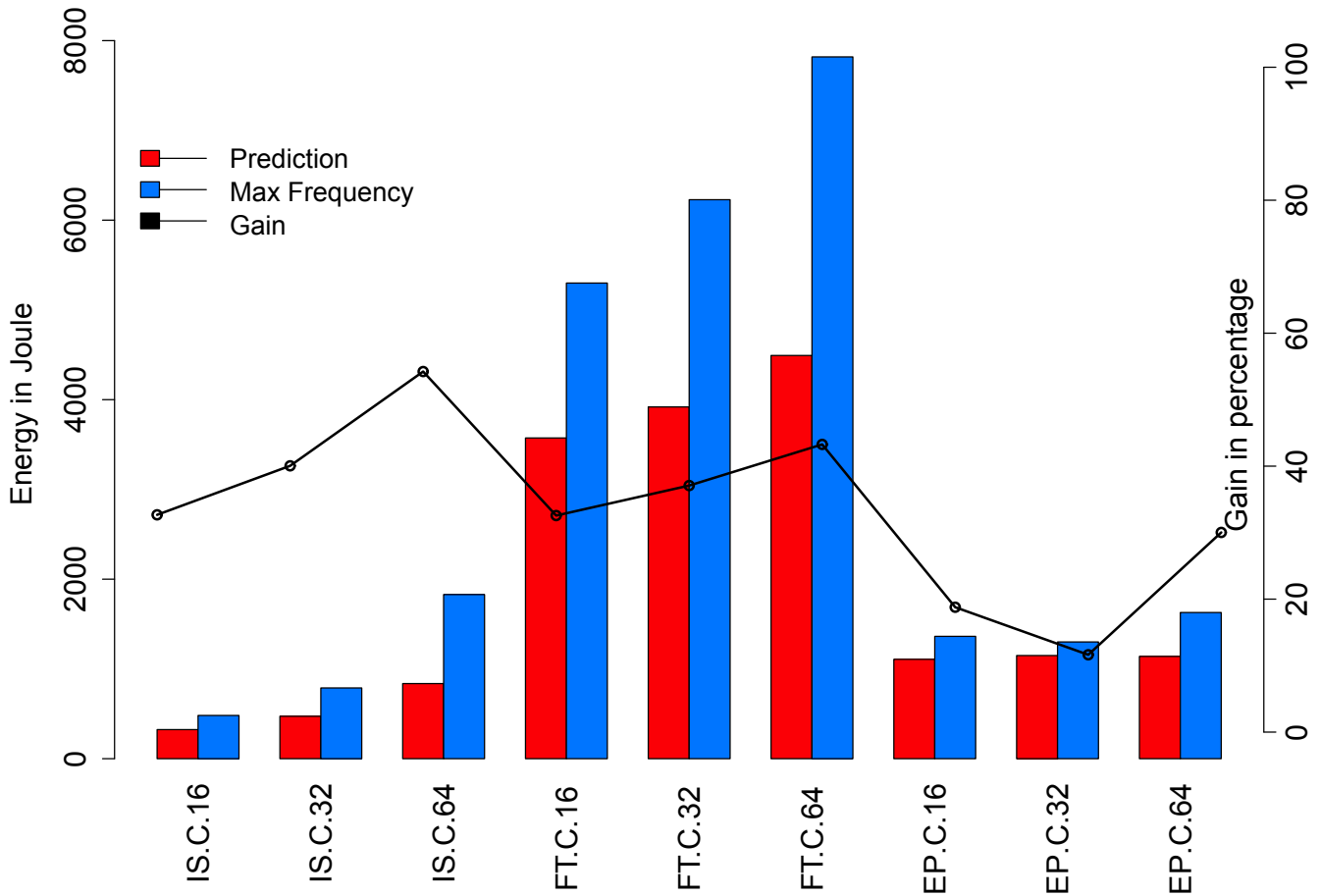


Figure : Possible energy gain

Outline

Conclusion and current work

- Conclusion
  - First attempt to offline scheduling
    - Linear programming problem taking into account architecture constraints
    - Processor-level tasks for a feasible solution

- Current work
  - Power and execution time prediction
  - Handling frequency transition overhead by grouping super tasks
  - Iterative solution