Speed Scaling for Energy and Performance with Instantaneous Parallelism

Hongyang Sun (Nanyang Technological University, Singapore)

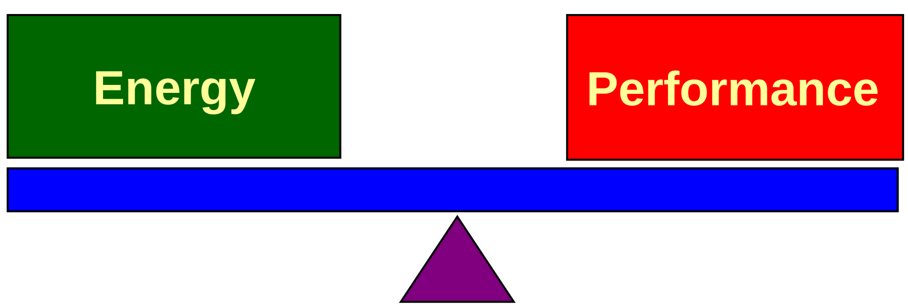
Joint work with

Wen-Jing Hsu (Nanyang Technological University, Singapore)

Yuxiong He (Microsoft Research, USA)

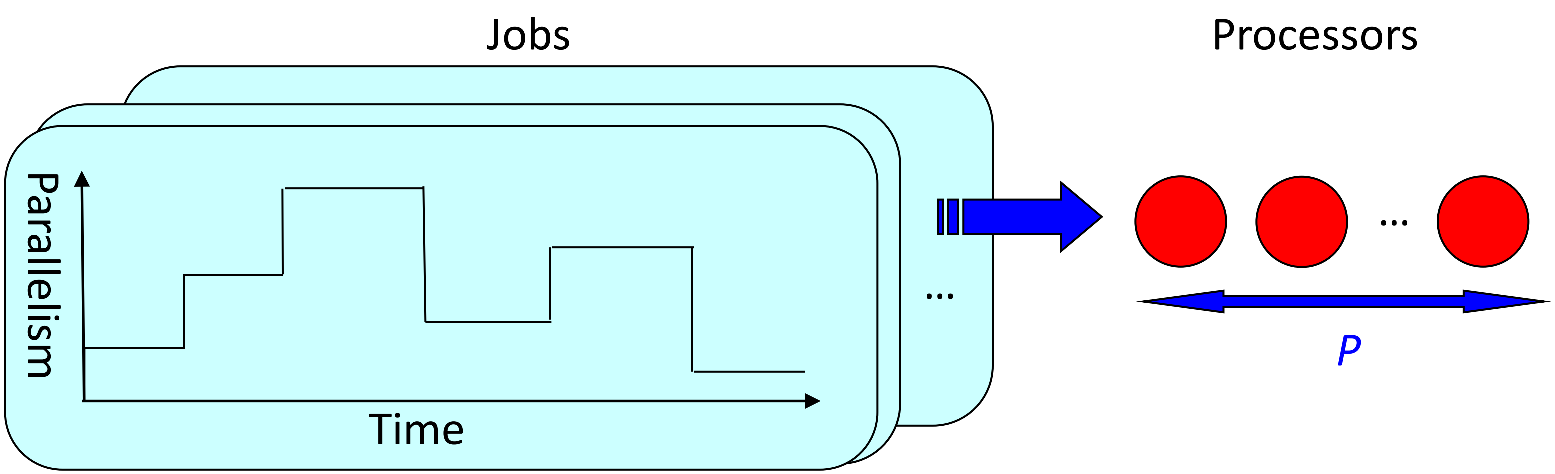
Background

- *Performance* has always been an important consideration.
- *Energy* has recently become a major concern as well.
- *Challenge*: How to achieve a balance/tradeoff between the two conflicting objectives.



Scheduling Problem

- Schedule a set of n jobs with *time-varying* parallelism on a set of P processors with *dynamic speed scaling (DVFS)* capability, assuming $n < P$.
  - ***Problem:*** Decide *online* how many processors to allocate to each job at any time and at what speeds?



Objectives

- Linear combination of performance and energy
  - *Performance*: sum or max of the jobs' execution time
    - Total flow time: sum of duration between release and completion of all jobs.
    - Makespan: maximal completion time of all jobs.
  - *Energy*: power of all processors integrated over time
    - A processor consumes power s^{α} when running at speed s, where $s \geq 0$ and $\alpha > 1$.
- Competitive analysis
  - An *online algorithm* is *c-competitive* if its cost is at most *c times* that of the *optimal offline algorithm*.

Different Degrees of Clairvoyance

- Non-clairvoyant
  - Knows *nothing* about the jobs, even the completed portions.
- Past-clairvoyant
  - Knows *past characteristics* of the jobs, i.e., the completed portions.
- IP-clairvoyant
  - Knows *instantaneous parallelism* (IP) of the jobs at any time.
- Semi-clairvoyant
  - Knows *approximate future* characteristics, but not exact ones.
- (Total)-Clairvoyant
  - Knows *everything* about the jobs, even the future characteristics.

Total Flow Time plus Energy

- Denoted by $G = F + E$.
- *Flow time* of a job is the duration between the job's release and completion.

Total Flow Time plus Energy

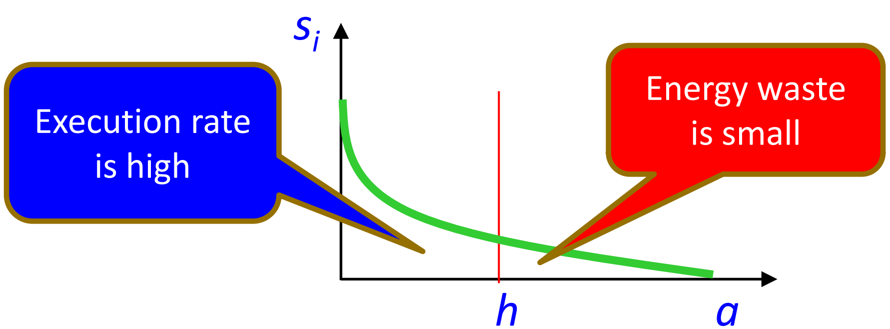
- Flow time of all jobs + Total energy of all jobs
  - First studied by [Albers&Fujiwara 2006].
  - More than a dozen papers have focused on this objective.
  - Most considered *sequential jobs* on uniprocessor or multiprocessors
    - *Challenge:* when and where to execute a job at what speed.
    - Single processor: clairvoyant: *2-comp.* [Andrew et al. 2009]; non-clairvoyant: $O(\alpha^2/\ln\alpha)$*-comp.* [Chan et al. 2009].
    - Multiprocessor: $O(1)$*-comp.* algorithms for clairvoyant [Lam et al. 2009] and non-clairvoyant [Greiner et al. 2009] settings.
  - Fewer results considered *parallel jobs* on multiprocessors
    - *Challenge:* How many processors for a job and at what speeds
    - Non-clairvoyant: $O(\ln^{1/\alpha} P)$*-comp.* for batched jobs [Sun et al. 2009]; $O(\ln P)$*-comp.* for non-batched jobs; $O(\log P)$*-comp.* for non-batched jobs [Chan et al. 2009] under a different execution model

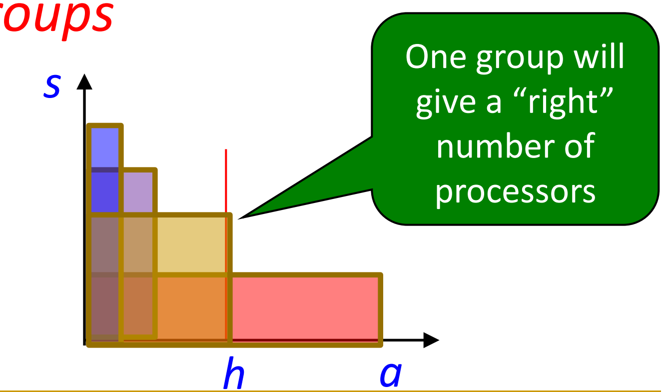
Total Flow Time plus Energy

- Techniques and Approaches
  - Balance *energy and flow* at any time
    - *Total power consumption u_t = number of active jobs n_t*
  - *Amortized local competitiveness argument* with the help of a potential function $\Phi(t)$
    - *Boundary condition*: $\Phi(0) = \Phi(\infty) = 0$
    - *Discrete-event condition*: $\Phi(t)$ should not increase at any discrete event, such as job arrival or completion.
    - *Running condition*: $dG_{ALG}(t)/dt + d\Phi(t)/dt \leq c \cdot dG_{OPT}(t)/dt$
    - $\rightarrow$ $G_{ALG} \leq c \cdot G_{OPT}(t)$, so ALG is *c-competitive.*
  - *Non-uniform* speed scaling (non-clairv. algorithms & parallel jobs)
    - Processors dedicated to a job can run at *different* speeds.
    - Uniform speed scaling was shown to be $\Omega(P^{(\alpha-1)/\alpha^2})$*-comp.*

Two Non-clairvoyant Execution Models

- Our Model
  - Processors of *different speeds* are given to a job; assume $s_1 \geq s_2 \geq \ldots \geq s_a$
  - Execution rate for the job at any time follows *maximum utilization policy*, i.e., *utilize faster processors first*
- Model by Chan,Edmonds,Pruhs
  - *Multiple processor groups* are given to a job; processors in same group share same speed, but can be different for different groups
  - Execution rate for the job at any time follows *maximum rate from all groups*





Upper Bounds and Lower Bounds

- Our Model
  - **Upper Bound**
    - N-EQUI (Non-clairvoyant)
      - EQUI: give $a = P/n_t$ processors

Identical asymptotic upper bounds and lower bounds in two models

$1/(l \cdot H_P)^{1/\alpha}$, where H_P is the P-th Harmonic number.

    - N-EQUI is $O(\ln P)$*-comp.*; $O(\ln^{1/\alpha} P)$*-comp.* (batched jobs)
  - **Lower Bound**
    - Any non-clairvoyant algorithm is $\Omega(\ln^{1/\alpha} P)$*-comp.*

- Model by Chan,Edmonds,Pruhs
  - **Upper Bound**
    - MultiLAPS (Non-clairvoyant)
      - LAPS: give $a = P/(\beta n_t)$ processors to
        ...al.
        ...are
        ...ps
        with geometrically decreasing size
      - Speed of each group $\approx 1/size^{1/\alpha}$
    - MultiLAPS is $O(\log P)$*-comp.*; $O(\log^{1/\alpha} P)$*-comp.* (batched jobs)
  - **Lower Bound**
    - Any non-clairvoyant algorithm is $\Omega(\log^{1/\alpha} P)$*-comp.*

An IP-clairvoyant Algorithm

- IP-clairvoyant
  - Knowing a job's instantaneous parallelism at any time
  - *No energy waste* since no excessive processor allocation
  - *Uniform speed scaling* should be sufficient
- U-CEQ (IP-clairvoyant)
  - CEQ: gives $a = \min\{h_t, P/n_t\}$ processors to each job
    - h_t is the instantaneous parallelism of the job at time t
    - P/n_t is the equal processor share for the job at time t
  - Uniform: set same speed for all a processors to $s = 1/a^{1/\alpha}$

Upper Bound of U-CEQ

- ***Theorem.*** U-CEQ *is O(1)-competitive for total flow time plus energy.*

  Proof sketch. Use potential function by [Lam et al. 2008]

$$\Phi(t) = \eta \int_0^\infty \left[\left(\sum_{i=1}^{n_t(z)} i^{1-1/\alpha} \right) - n_t(z)^{1-1/\alpha} n_t^*(z) \right] dz,$$

  - Guarantees *boundary* and *discrete-event* conditions.
  - For *running* condition, we can show $c = \max\{\frac{2\alpha^2}{\alpha-1}, 2^\alpha \alpha\}$

- ***Remarks.*** *Competitive ratio is independent of P, but depends on α. Maybe more future info can help.*

Makespan plus Energy

- Denoted by $H = M + E$.
- *Makespan* of a job set is the completion time of the last completed job.

Makespan plus Energy

- Time last job completes + Total energy of all jobs
  - *Last job* contributes to *both makespan and energy.*
  - *Other jobs* contribute only to *energy* → can be slowed down to save energy w/o affecting makespan.
  - *Challenge:* which jobs to slow down in non-clairvoyant, or even IP-clairvoyant setting?
  - *Intuition:* without knowing future info., treating all jobs equally by running them at same rate.

Constant Power Property

Lemma. *To minimize makespan plus energy for batched jobs, power at any time should be constant at $1/(\alpha-1)$.*

Proof sketch. Balance power and makespan at any time. Suppose that power at some time is not $1/(\alpha-1)$, then either speeding up or slowing down the execution of all jobs will lead to smaller overall cost.

Remark 1. *For nonbatched jobs, slowing down still works, but speeding up doesn't → power should be $\leq 1/(\alpha-1)$.*

Remark 2. *To minimize total flow time plus energy for batched job, power should be constant at $n_t/(\alpha-1)$.*

- Scaling MultiLAPS achieves $O(\log^{1/\alpha} P)$-*comp.* for batched jobs.

An IP-clairvoyant Algorithm

- Parallel-First
  - Applicable to *batched (Par-Seq)** jobs, i.e., with fully-parallelizable and sequential phases.
  - Run any fully-parallel phase from any job whenever possible, using all processors with same speed;
  - Otherwise, run sequential phases of all jobs *at the same rate*, each on one processor.
- Idea can be generalized to scheduling jobs with *arbitrary parallelism profile* → *same asymptotic bound*
  - Run all processors whenever possible with the same speed;
  - Otherwise, run all jobs at the same power.

Performance of Parallel-First

Theorem. *Parallel-First is $\Theta(\ln^{1-1/\alpha}P)$-competitive for makespan plus energy of batched (Par-Seq)* jobs.*

Proof. Consider difference between OPT and PF

- Fully-parallelizable phases: OPT and PF execute same way
- Sequential phases: $w_1 \le w_2 \le \ldots \le w_P$
  - OPT: finish all jobs simultaneously → $H^* = \Theta(\sum_{i=1..P} w_i^{\alpha})^{1/\alpha}$
  - PF: execute all jobs at same speed → $H = \Theta(\sum_{i=1..P} w_i/(P-i+1)^{1-1/\alpha})$
  - Maximize with *Lagrange multiplier* → $H/H^* = O(\ln^{1-1/\alpha}P)$
  - Suppose that $w_i = 1/(P-i+1)^{1/\alpha}$ → $H/H^* = \Omega(\ln^{1-1/\alpha}P)$

Lower Bound for IP-Clairvoyant Algo.

Theorem. *Any IP-clairvoyant algorithm is $\Omega(\ln^{1-1/\alpha} n)$-competitive for makespan plus energy.*

Proof sketch. Any IP-clairvoyant algorithm that does not execute sequential phases using same speed will cost more than PF in worst case, as adversary can always make it assign "wrong" speeds to jobs, e.g., assign faster processors to shorter jobs.

Question. *What is competitiveness lower bound for non-clairvoyant algorithms, which can also assign a "wrong" number of processors to jobs?*

Comparing Makespan and Total Flow

	Non-clairvoyant	IP-clairvoyant
Total Flow + Energy	$\Omega(\ln^{1/\alpha} P)$-competitive	$O(1)$-competitive
Makespan + Energy	?	$\Omega(\ln^{1-1/\alpha} P)$-competitive

Minimizing “makespan + energy” seems more challenging than minimizing “total flow time + energy”

Final Remark

- Parallel job models
  - Used here: parallelism profile model
    - Each phase has a *linear* speedup function up to some parallelism
  - More general: Edmonds' model
    - Each phase can have *non-decreasing* and *sub-linear* speedup
    - Reduced to parallelism profile for non-clairvoyant algorithms
    - Reduction for or design of IP-clairvoyant algorithms?

Thank you!