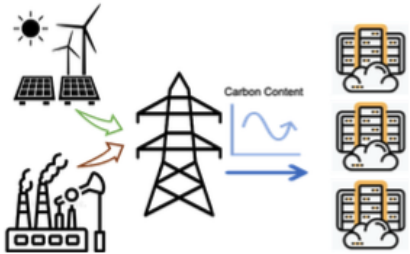
Variable Capacity Scheduling

Anne Benoit
LIP, Ecole Normale Supérieure de Lyon
Institut Universitaire de France

Joint work with Y. Robert, L. Perotin, J. Cendrier, F. Vivien (ENS Lyon)
and A. A. Chien, R. Wijayawardana, C. Zhang (U. Chicago)

September 4, 2024 – CCDSC – La Maison des Contes

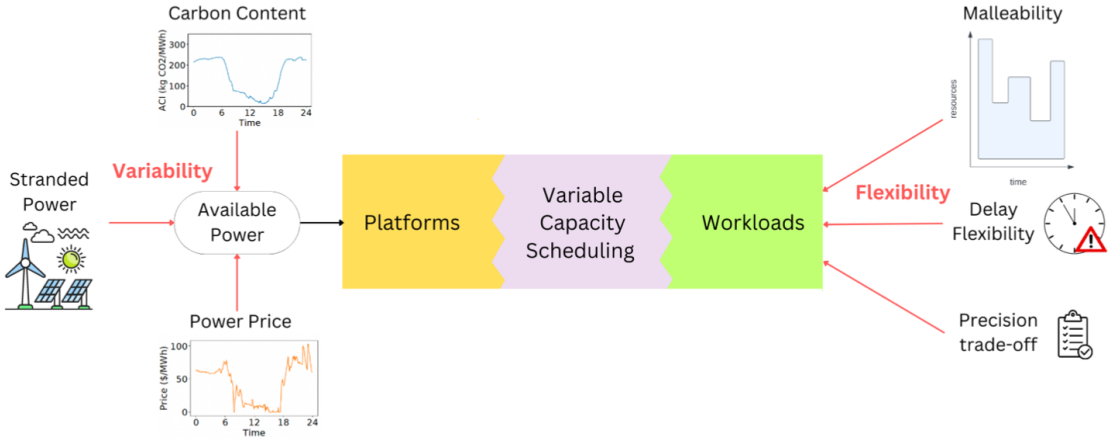
Variable power



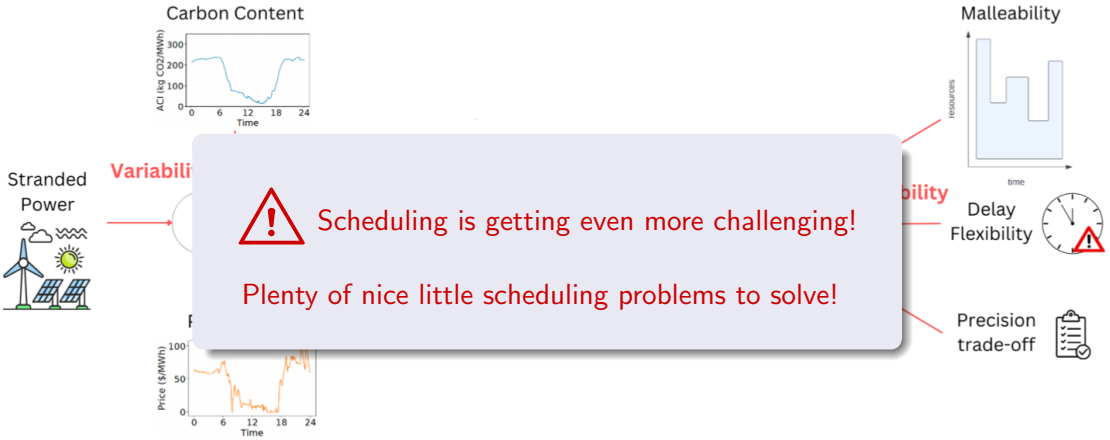
- Today's data centers assume resource capacity as a fixed quantity
- Emerging approaches:
  - Exploit grid renewable energy
  - Reduce carbon emissions

$\Rightarrow$ Variable power

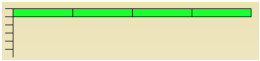
Big picture


Carbon Content
ACI (kg CO2/MWh)
300
200
100
0
0 6 12 18 24
Time
Malleability
resources
time
Stranded Power
Variability
Available Power
Platforms
Variable Capacity Scheduling
Workloads
Flexibility
Delay Flexibility
Precision trade-off
Power Price
Price ($/MWh)
100
50
0
0 6 12 18 24
Time

Big picture


Carbon Content
ACI (kg CO2/MWh)
300
200
100
0
0 6 12 18 24
Time
Malleability
resources
time
Stranded
Power
Variabili
bility
Delay
Flexibility
Precision
trade-off
Price ($/MWh)
100
50
0
0 6 12 18 24
Time
Scheduling is getting even more challenging!
Plenty of nice little scheduling problems to solve!

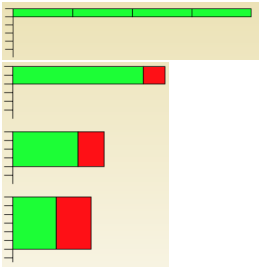
Parallel jobs



- **Rigid jobs**: Processor allocation is fixed
- **Moldable jobs**: Processor allocation is decided by the user or the system but cannot be changed during execution
- **Malleable jobs**: Processor allocation can be dynamically changed

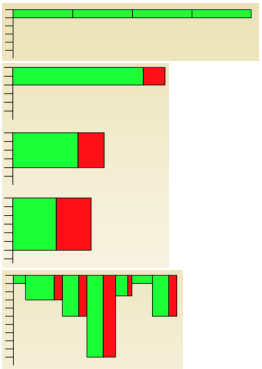
Parallel jobs

- **Rigid jobs**: Processor allocation is fixed
- **Moldable jobs**: Processor allocation is decided by the user or the system but cannot be changed during execution
- **Malleable jobs**: Processor allocation can be dynamically changed



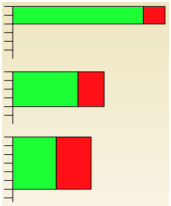
Parallel jobs

- **Rigid jobs**: Processor allocation is fixed
- **Moldable jobs**: Processor allocation is decided by the user or the system but cannot be changed during execution
- **Malleable jobs**: Processor allocation can be dynamically changed



Parallel jobs

- **Rigid jobs**: Processor allocation is fixed



- **Moldable jobs**: Processor allocation is decided by the user or the system but cannot be changed during execution

- **Malleable jobs**: Processor allocation can be dynamically changed

The case for moldable jobs:

- Easily adapt to the amount of available resources (contrarily to rigid jobs)
- Easy to design/implement (contrarily to malleable jobs)
- Computational kernels in scientific libraries are provided as moldable jobs

Checkpoints

- Some jobs cannot be interrupted
- Some jobs can be checkpointed

Half the projected load for US Exascale systems include checkpointing capabilities
(from APEX worklows, Sandia/LosAlamos/NERSC report, April 2016)

Checkpoints

Scheduling opportunity

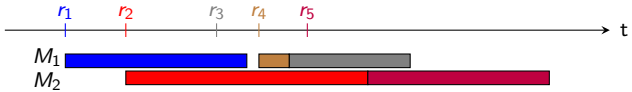
- Many checkpointable jobs are moldable
- These jobs are able to restart with a different allocation (size and shape)

Resizing impacts performance

(from APEX workflows, Sandia/LosAlamos/NERSC report, April 2016)

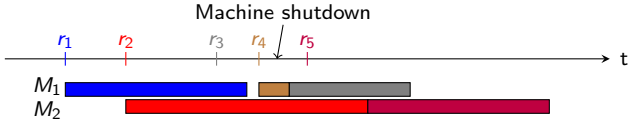
Risk aware?

1. **Which machine to shutdown?**



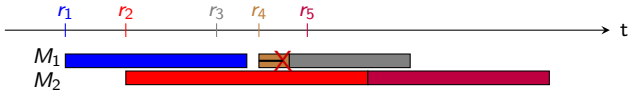
Risk aware?

1. **Which machine to shutdown?**



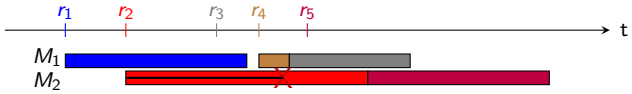
Risk aware?

1. **Which machine to shutdown?**



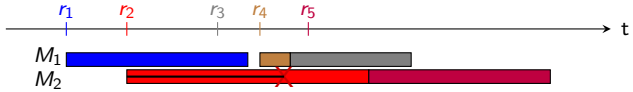
Risk aware?

1. **Which machine to shutdown?**



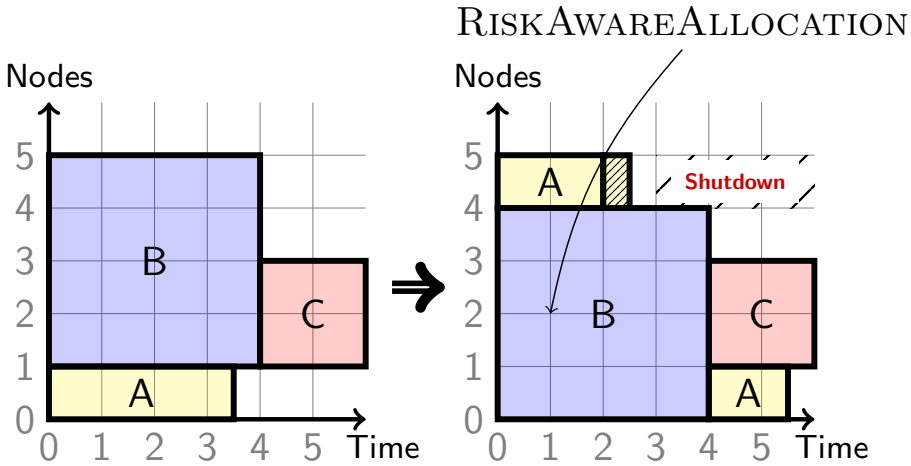
Risk aware?

1. **Which machine to shutdown?**



2. **How to schedule jobs to minimize impact?**

Small example



Main questions

- When power decreases, which machines to power off? Which jobs to interrupt? And to re-schedule?
- Are we notified ahead of a power change?
  - Resource variation in power obeys specific parameters whose evolution is dictated by a mix of technical availability and economic conditions
  - Accurate external predictor (precision, recall)? Maybe too optimistic ☹
- Re-scheduling interrupted jobs
  - Can we take a proactive checkpoint before the interruption?
  - Which priority should be given to each interrupted job?
  - Which geometry and which nodes for re-execution?

Main questions

- When power decreases, which machines to power off? Which jobs to interrupt? And to re-schedule?
- Are we notified ahead of a power change?

Scheduling opportunity & challenge

- Nodes ordered according to non-decreasing risk, say from left to right
- Shutdown nodes starting from the right
- Assign priority jobs, such as large jobs, to nodes on the left
- Global load of the platform must remain balanced

Sophisticated algorithms that go well beyond first-fit decisions

Outline

Framework

- Set of rigid **jobs**, each using a given number of cores (work w_i on c_i cores)
- Identical multicore **machines**, number of machines alive evolves with time
- Number of alive machines not known until it changes
- No possibility to checkpoint jobs or to anticipate a resource variation
- **Objective function:** Goodput $\Rightarrow$ fraction of useful work up to time T
  - $\mathcal{J}_{comp,T}$: set of jobs that are complete at time T ($e_i \leq T$)
  - $\mathcal{J}_{started,T}$: set of jobs running and not finished at time T ($s_i \leq T < e_i$)
  - Total number of units of work that can be executed in $[0, T]$: $n_c \sum_{t \in [0,T-1]} M_{alive}(t)$

$$\textsc{Goodput}(T) = \frac{\sum_{\tau_i \in \mathcal{J}_{comp,T}} w_i c_i + \sum_{\tau_i \in \mathcal{J}_{started,T}} (T - s_i) c_i}{n_c \sum_{t \in [0,T-1]} M_{alive}(t)}$$

Keep an eye on maximum stretch

Framework

- Set of rigid **jobs**, each using a given number of cores (work w_i on c_i cores)
- Identical multicore **machines**, number of machines alive evolves with time
- Number of alive machines not known until it changes
- No possibility to checkpoint jobs or to anticipate a resource variation
- **Objective function:** Goodput $\Rightarrow$ fraction of useful work up to time T
  - $\mathcal{J}_{comp,T}$: set of jobs that are complete at time T ($e_i \leq T$)
  - $\mathcal{J}_{started,T}$: set of jobs running and not finished at time T ($s_i \leq T < e_i$)
  - Total number of units of work that can be executed in $[0, T]$: $n_c \sum_{t\in[0,T-1]} M_{alive}(t)$

$$\textsc{Goodput}(T) = \frac{\sum_{\tau_i\in\mathcal{J}_{comp,T}} w_i c_i + \sum_{\tau_i\in\mathcal{J}_{started,T}} (T - s_i)c_i}{n_c \sum_{t\in[0,T-1]} M_{alive}(t)}$$

Keep an eye on maximum stretch

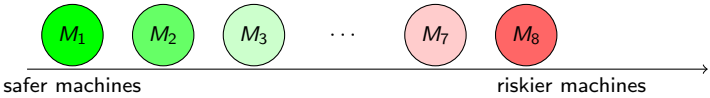
Framework

- Set of rigid **jobs**, each using a given number of cores (work w_i on c_i cores)
- Identical multicore **machines**, number of machines alive evolves with time
- Number of alive machines not known until it changes
- No possibility to checkpoint jobs or to anticipate a resource variation
- **Objective function:** Goodput $\Rightarrow$ fraction of useful work up to time T
  - $\mathcal{J}_{comp,T}$: set of jobs that are complete at time T ($e_i \leq T$)
  - $\mathcal{J}_{started,T}$: set of jobs running and not finished at time T ($s_i \leq T < e_i$)
  - Total number of units of work that can be executed in $[0, T]$: $n_c \sum_{t\in[0,T-1]} M_{alive}(t)$

$$\textsc{Goodput}(T) = \frac{\sum_{\tau_i\in\mathcal{J}_{comp,T}} w_i c_i + \sum_{\tau_i\in\mathcal{J}_{started,T}} (T - s_i)c_i}{n_c \sum_{t\in[0,T-1]} M_{alive}(t)}$$

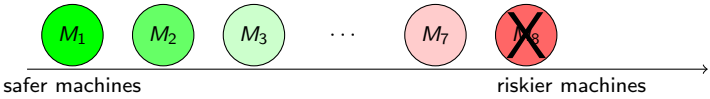
Keep an eye on maximum stretch

Risk-aware



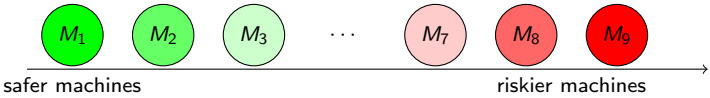
Risk-aware job allocation strategies

Risk-aware



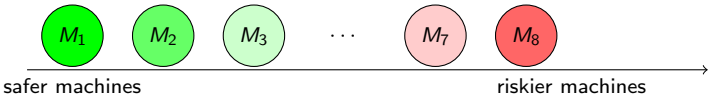
Risk-aware job allocation strategies

Risk-aware



Risk-aware job allocation strategies

Risk-aware



Risk-aware job allocation strategies

Events:

- **Job arrival**: When a job is released, when to schedule it and on which machine?
- **Job completion**: When a job is completed, its cores are released $\Rightarrow$ additional jobs can be scheduled
- **Machine addition**: When a new machine becomes available, how to utilize it?
- **Machine removal**: When a machine is turned off, its jobs are killed and need re-allocation

Heuristics

- FirstFitAware:
  - Ordered list of machines
  - Jobs mapped to leftmost (safer) machines whenever possible
  - Rightmost (riskier) machines are shutdown whenever necessary
- FirstFitUnaware: Shutdown random machines whenever necessary
- Can we do better than first fit?
  - Interrupting a long job is a big performance loss
  - Schedule smaller jobs on machines that are likely to be turned off
  - Schedule longer jobs on risk-free machines

Heuristics

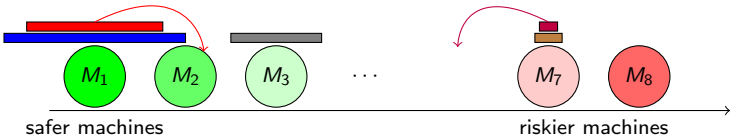
- FirstFitAware:
  - Ordered list of machines
  - Jobs mapped to leftmost (safer) machines whenever possible
  - Rightmost (riskier) machines are shutdown whenever necessary
- FirstFitUnaware: Shutdown random machines whenever necessary
- Can we do better than first fit?
  - Interrupting a long job is a big performance loss
  - Schedule smaller jobs on machines that are likely to be turned off
  - Schedule longer jobs on risk-free machines

TargetStretch, TargetASAP, & PackedTargetASAP

- TargetStretch: Add one queue per machine, target value for max stretch
  potential bad utilization
  No flexibility for mapping to another free machine

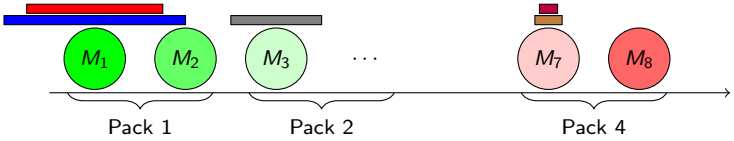
TargetStretch, TargetASAP, & PackedTargetASAP

- TargetStretch: Add one queue per machine, target value for max stretch
  potential bad utilization
  No flexibility for mapping to another free machine
- TargetASAP:
  - Start job immediately on target machine or closest machine in neighborhood
  - If not possible, assign on target machine if target stretch not exceeded
  - Otherwise, assign on machine where it can start ASAP (within acceptable distance)



TargetStretch, TargetASAP, & PackedTargetASAP

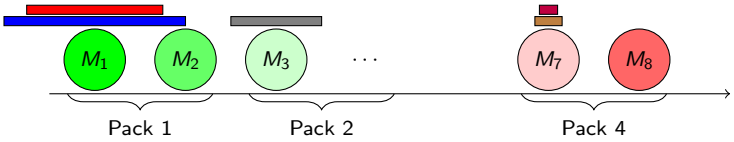
- TargetStretch: Add one queue per machine, target value for max stretch
  potential bad utilization
  No flexibility for mapping to another free machine
- TargetASAP:
  - Start job immediately on target machine or closest machine in neighborhood
  - If not possible, assign on target machine if target stretch not exceeded
  - Otherwise, assign on machine where it can start ASAP (within acceptable distance)
- Variant PackedTargetASAP: group machines into packs, and assign jobs to first machines of the pack, to leave machines empty for future large jobs



TargetStretch, TargetASAP, & PackedTargetASAP

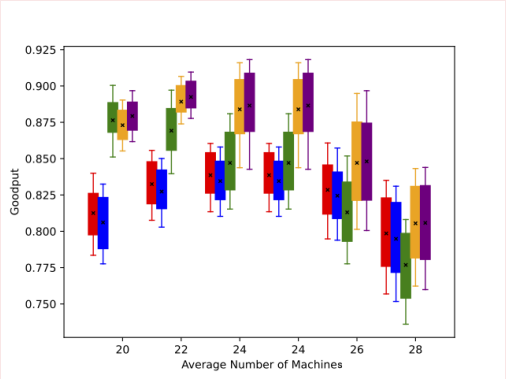
- TargetStretch: Add one queue per machine, target value for max stretch
  potential bad utilization
  No flexibility for mapping to another free machine
- TargetASAP:
  - Start job immediately on target machine or closest machine in neighborhood
  - If not possible, assign on target machine if target stretch not exceeded
  - Other… table distance)
- Variant P… ssign jobs to first machines of the pack, to leave machines empty for future large jobs

Technical and kind of painful
despite all simplifying hypotheses ☹



TargetStretch, TargetASAP, & PackedTargetASAP

Simulation results using resource variation trace and job traces (Borg)
Significant gains over first-fit algorithms: map the right job to the right machine



Outline

Model

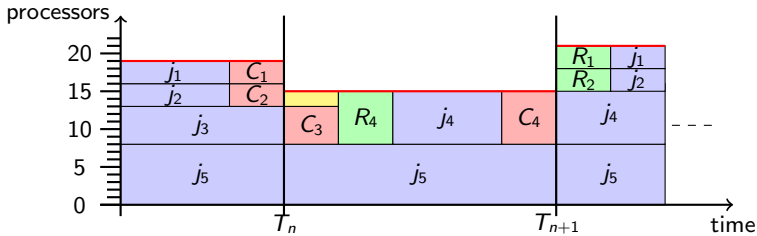
Problem: Scheduling infinite parallel rigid jobs under variable number of processors, during each *section*

Hypotheses:

- A job can be **checkpointed** and recovered
- Knowledge of the duration of each section, and bound on #proc difference

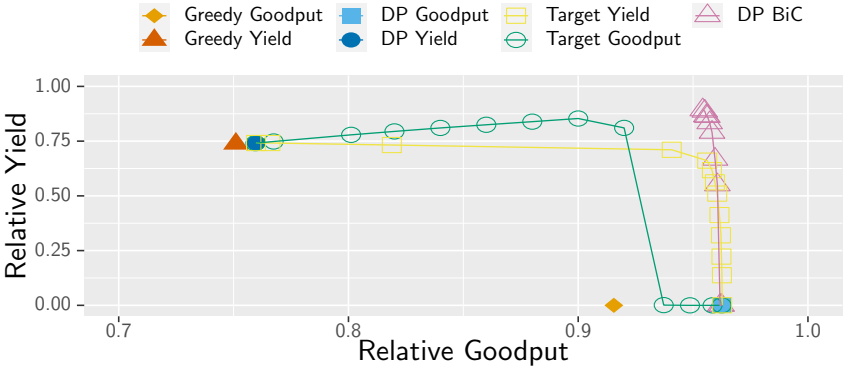
Additional constraint:

- Never lose work (i.e., checkpoint enough before section change, and never shut off a non-checkpointed job)



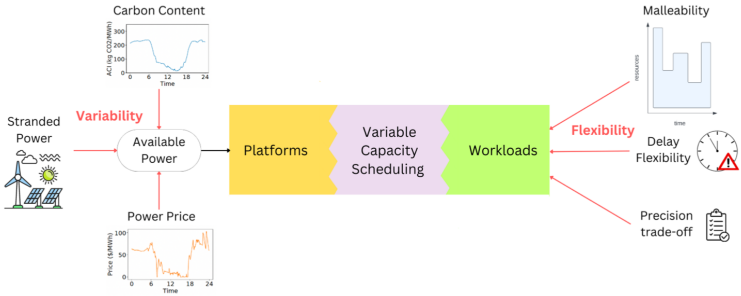
Algorithms

- Sophisticated dynamic programming algorithms to optimize goodput and/or yield at the end of a section
- Evaluation on job traces
- Improvement of novel strategies over greedy approaches



Outline

Back to the big picture



Many challenging scheduling problems when resources subject to variable capacity ☺

Workshop report: *Scheduling Variable Capacity Resources for Sustainability*; March 29-31, 2023, U. Chicago Paris Center

Today's case studies: restricted instances ☹

Risk-Aware Scheduling Algorithms for Variable Capacity Resources; PMBS workshop at SC'23

Research directions

Platforms and resources: New and more complex definitions of capacity; understand and model capacity changes

Flexible workloads: Exploit flexible start dates, allow migration or deferral, support multiple precision levels

Scheduling models and metrics: Consider new multi-criteria metrics for both performance and sustainability (including carbon cost); Account for uncertainty

Policy and societal factors: Mechanisms that help people accept constraints linked to environmental rules; Beware of the superficial feeling of abundance: abuse of computational resources, rebound effect

Research directions

Platforms anity;
understand an

Flexible workerral, support
multiple precis

Scheduling mfor both
performance a rtainty

Policy and sostraints linked
to environmen buse of
computational resources, rebound effect

Plenty of nice little scheduling problems to solve in the near future!

Happy to discuss around a drink... ☺

Thanks again for the invitation to this great workshop!