Speculative Scheduling for Stochastic HPC Applications

Guillaume Pallez
Inria

M2 CISD, Enseirb-Matmeca,
Automne 2020

based on a work with Gainaru, Raghavan and Sun

Plan du cours d'aujourd'hui

1. Motivation
   - Batch scheduling
   - Stochastic Apps
2. Model
   - Job model
   - Platform model and Optimization objective
3. Algorithm
   - High-level
4. Evaluation
   - Evaluation framework
   - Different scenarios
5. Concl, perspectives

Plan

HPC schedulers

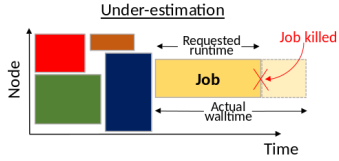
Reservation-based batch schedulers:

- Relies on (reasonably) accurate runtime estimation from the user/application
- Two queues: (i) large (main) jobs; (ii) small jobs used for backfilling.
- **Cost to users:** Pay what you use. $\rightarrow$ need to guarantee that the time asked is sufficient.

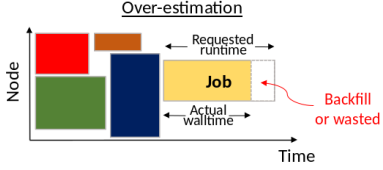
HPC schedulers

Reservation-based batch schedulers:

- Relies on (reasonably) accurate runtime estimation from the user/application
- Two queues: (i) large (main) jobs; (ii) small jobs used for backfilling.
- **Cost to users:** Pay what you use. → need to guarantee that the time asked is sufficient.



- Job killed, need to resubmit; additional cost to user.



- Job completed early (?).
- May waste system resources.

Motivational examples

Sysadmin: “I want to sell all the compute slots on my platform”

Motivational examples

Sysadmin: “I want to sell all the compute slots on my platform”

User: “ I don’t want to pay if I don’t use”

Motivational examples

Sysadmin: "I want to sell all the compute slots on my platform"

User: " I don't want to pay if I don't use"

Sysadmin: "Sure, then you only pay what you use."

Motivational examples

Sysadmin: "I want to sell all the compute slots on my platform"

User: " I don't want to pay if I don't use"

Sysadmin: "Sure, then you only pay what you use."

User has one job J_1 whose execution time is **exactly 50h**.

- What does User do?
- Is Sysadmin happy?

Motivational examples

Sysadmin: "I want to sell all the compute slots on my platform"

User: " I don't want to pay if I don't use"

Sysadmin: "Sure, then you only pay what you use."

User has one job J_1 whose execution time is **exactly 50h**.

- What does User do?
- Is Sysadmin happy?

User has one job J_2 whose execution time is **between 46h and 54h**.

- What does User do?
- Is Sysadmin happy?

Motivational examples

Sysadmin: "I want to sell all the compute slots on my platform"

User: " I don't want to pay if I don't use"

Sysadmin: "Sure, then you only pay what you use."

User has one job J_1 whose execution time is **exactly 50h**.

- What does User do?
- Is Sysadmin happy?

User has one job J_2 whose execution time is **between 46h and 54h**.

- What does User do?
- Is Sysadmin happy?

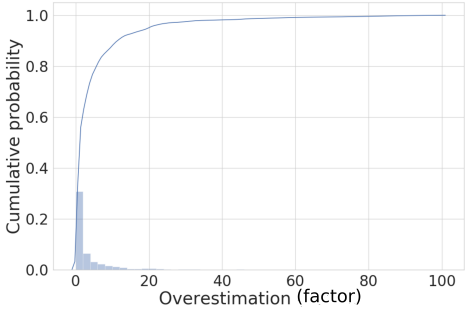
User has one job J_3 whose execution time is **between 2h and 98h**.

- What does User do?
- Is Sysadmin happy?

Anecdotal?

Study of application data from Intrepid (2009 ANL system) (data from Parallel Workload Archive).

Average job size	880 nodes / 3089 node hours
Average small jobs size	48.6 nodes / 31 node hours
Over-estimated submissions	82.2 %
Under-estimated submissions	17.7%
Average over-estimation space	2132 node hours
Percentage of small jobs	30.8%



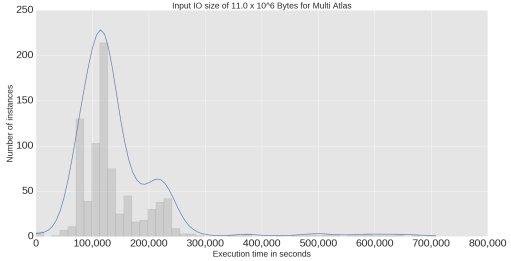
$\Longrightarrow$ **Unused backfilling space: 2.8 hours/day**

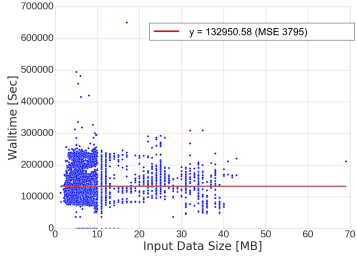
$$\text{factor} = \frac{\text{estimate - walltime}}{\text{walltime}}$$

Stochastic applications

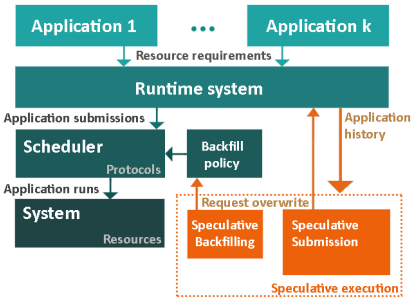
"Second generation" of HPC applications (BigData, ML) with heterogeneous, dynamic and data-intensive properties.

- Execution time is *input dependent*
- Unpredictable even for *same input size*
- Large variations





Contributions



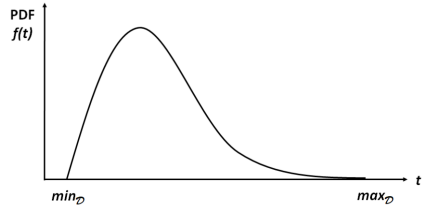
- Demonstrate the efficiency of using a multi-request type algorithm for HPC schedulers
  - Idea: Overwrite for all jobs their requested time at submission
- Demonstrate the efficiency of *Speculative backfilling*
  - Idea: Overwrite the request time temporarily during backfill

Plan

Stochastic Jobs

Job execution time follows a Random Variable X.

- Distribution $\mathcal{D}$
- Cumulative function (CDF) F ($F(x) = \mathbb{P}(X \leq x)$)*
- Density function (PDF) f
- Support is positive ($X \in [\min_{\mathcal{D}}, \max_{\mathcal{D}}]$, s.t. $\min_{\mathcal{D}} \geq 0$ and $\max_{\mathcal{D}} \in \mathbb{R} \cup \{\infty\}$)



- Deterministic jobs (two executions of the same job have the same duration).

*most of the results assume a smooth CDF.

Model

- A system with P identical processors and two queues.

Model

- A system with P identical processors and two queues.
- **Long queue:** $\mathcal{J} = \{J_1, J_2, \ldots, J_M\}$ of **large** stochastic jobs
  - processor allocation p_j
  - each walltime follows a given probability distribution (random variable)

Model

- A system with P identical processors and two queues.
- **Long queue:** $\mathcal{J} = \{J_1, J_2, \ldots, J_M\}$ of **large** stochastic jobs
  - processor allocation p_j
  - each walltime follows a given probability distribution (random variable)
- **Short queue:** A stream $\mathcal{B}$ of **small** jobs
  - arrival rate λ
  - average execution time ε much smaller than that of the large jobs.
  - **Continuous approximation:** modeled as a stream of work arriving continuously in the queue with a rate $\mathrm{Z} = \lambda\varepsilon$

Model

- A system with P identical processors and two queues.
- **Long queue:** $\mathcal{J} = \{J_1, J_2, \ldots, J_M\}$ of **large** stochastic jobs
  - processor allocation p_j
  - each walltime follows a given probability distribution (random variable)
- **Short queue:** A stream $\mathcal{B}$ of **small** jobs
  - arrival rate λ
  - average execution time ε much smaller than that of the large jobs.
  - **Continuous approximation:** modeled as a stream of work arriving continuously in the queue with a rate $\mathrm{Z} = \lambda\varepsilon$

Optimization objective

- System Utilization: Useful Work / (P·Total Time)
- System response time: average time between submission and completion.

Reservation-based Approach

Given a job J of duration t (unknown). The user makes a reservation of time t_1. Two cases:

- $t \leq t_1$ The reservation is enough and the job succeeds.
- $t > t_1$ The reservation is not enough. The job fails. The user needs to ask for another reservation $t_2 > t_1$.

A **strategy** is a sequence of such reservations.

Reservation-based Approach

Given a job J of duration t (unknown). The user makes a reservation of time t_1. Two cases:

- $t \leq t_1$ The reservation is enough and the job succeeds.
- $t > t_1$ The reservation is not enough. The job fails. The user needs to ask for another reservation $t_2 > t_1$.

A **strategy** is a sequence of such reservations.

For J_3 (exec **2h to 98h**):
• **Strategy:** $t_1 = 5h$, $t_2 = 40h$, $t_3 = 60h$, $t_4 = 98h$.

If the job is 33h:
1. We run the $5h$ reservation; it fails.
2. **Then** we run the $40h$; it succeeds.

Is the sysadmin happy?

Is the user happy?

Reservation-based Approach

Given a job J of duration t (unknown). The user makes a reservation of time t_1. Two cases:

- $t \leq t_1$ The reservation is enough and the job succeeds.
- $t > t_1$ The reservation is not enough. The job fails. The user needs to ask for another reservation $t_2 > t_1$.

A **strategy** is a sequence of such reservations.

For J_3 (exec **2h to 98h**):
• **Strategy:** $t_1 = 5h$, $t_2 = 40h$, $t_3 = 60h$, $t_4 = 98h$.

If the job is 33h:
1. We run the $5h$ reservation; it fails.
2. **Then** we run the $40h$; it succeeds.

Is the sysadmin happy?
Util: 33/45 instead of 33/98

Is the user happy?

Reservation-based Approach

Given a job J of duration t (unknown). The user makes a reservation of time t_1. Two cases:

- $t \leq t_1$ The reservation is enough and the job succeeds.
- $t > t_1$ The reservation is not enough. The job fails. The user needs to ask for another reservation $t_2 > t_1$.

A **strategy** is a sequence of such reservations.

For J_3 (exec **2h to 98h**):
• **Strategy:** $t_1 = 5h,\ t_2 = 40h,\ t_3 = 60h,\ t_4 = 98h$.

If the job is 33h:
1. We run the $5h$ reservation; it fails.
2. **Then** we run the $40h$; it succeeds.

Is the sysadmin happy?
Util: 33/45 instead of 33/98

Is the user happy?
Cost: 38 instead of 33.

Plan

Two phase scheduling algorithm

Truthfully I do not know how to maximize the expected utilization. Writing the problem is already painful.

Instead we'll go naive with a two phase algorithm based on intuition:

- First phase: compute a reservation strategy for each job J_i: $\{t_{i,1}, t_{i,2}, \dots\}$.
- Second phase: reservation scheduling

Phase 1: Reservation strategy

Idea: Use the reservation strategy that minimizes the expected makespan (**TOptimal**) as if job J_i was alone in the system (more details: Aupy et al., IPDPS'19)

- It is optimal for utilization if job J_i is the only large job in the system ☺.
- We extended it (**ATOptimal**) to take into account backfilling: we define for J_i its backfilling rate:

$$\zeta_i = \mathrm{Z} \cdot \frac{p_i}{P} = \lambda\varepsilon\frac{p_i}{P}$$

Phase 1: Reservation strategy

Idea: Use the reservation strategy that minimizes the expected makespan (**TOptimal**) as if job J_i was alone in the system (more details: Aupy et al., IPDPS'19)

- It is optimal for utilization if job J_i is the only large job in the system ☺.
- We extended it (**ATOptimal**) to take into account backfilling: we define for J_i its backfilling rate:

$$\zeta_i = \mathrm{Z} \cdot \frac{p_i}{P} = \lambda\varepsilon\frac{p_i}{P}$$

Algorithm	**Sequence of requests (in hours)**
TOptimal	10.8, 13.4, 15.4, 17.1, 18.7, 20.0
ATOptimal ($\zeta = 0.1$)	10.86, 13.91, 18.69, 20.0
ATOptimal ($\zeta = 0.5$)	13.04, 20.0
ATOptimal ($\zeta = 0.9$)	17.39, 20.0
ATOptimal ($\zeta = 1$)	20.0

Example of strategies depending on the backfilling rate ζ.
Distribution is Truncated Normal on 0 to 20 hours, $\mu = 8$h, $\sigma = 2$h

Phase 2: Job scheduling

We follow a batch scheduler model. We want to execute a batch of jobs from the long queue (typically 100 jobs).

1. For all jobs of the batch, submit to the scheduler their smallest reservation ($\forall i,\ t_{i,1}$).
2. Let the scheduler compute its schedule the usual way
3. In case of $t_{i,1}$ is not enough, J_i is resubmitted with $t_{i,2}$
4. The scheduler computes a new schedule with all resubmitted $t_{i,2}$ and so on.

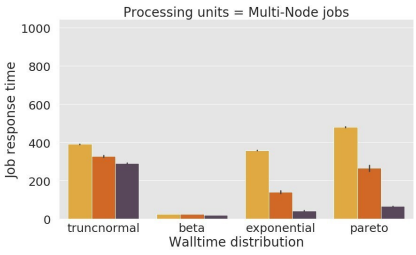
Plan

Evaluations

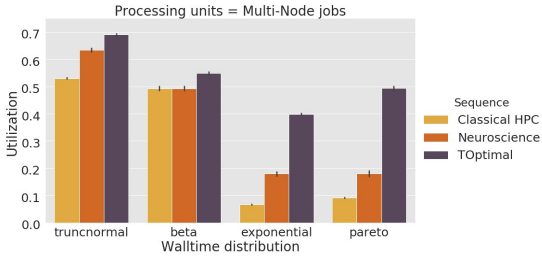
Four scenarios:

- Scenario 1: No backfilling. Jobs are represented by different probability distribution (both for execution time and number of processors).
- Scenario 2: Inclusion of backfilling jobs whose execution time is known.
- Scenario 3: Backfilling jobs whose execution time is unknown. *Speculative backfilling*.
- Scenario 4: Instantiation with Intrepid parameters (platform); neuroscience applications (jobs). Evaluation on two weeks simulation.

Scenario 1: no backfilling jobs



(a) Average job response time



(b) System utilization

System utilization and average job response time under different walltime distributions for jobs whose processor allocations follow the Beta distribution

Neuroscience uses the last few runs to decide the requested time and 1.5x increase factor in case of failures

Scenario 2: with known backfilling jobs

Large Jobs

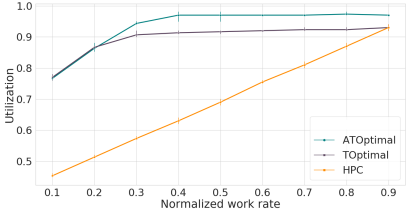
- Identical execution time profile: Truncated Normal distribution between 1 to 20h. Mean execution time: 8h. Variance: 2h.

Backfilling jobs

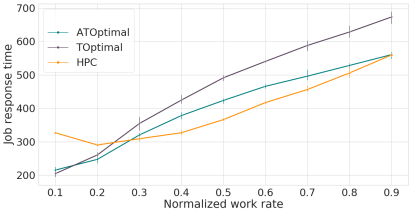
- Discrete jobs, generated with expected execution time 100× smaller than that of large jobs.
- Arrival rate through time to match the desired value for Z (=Normalized work rate).
- For backfilling purpose, we assume we know their exact execution time.

Scenario 2: with known backfilling jobs

- Results for ATOptimal move between TOptimal ($\zeta = 0$) and HPC
- The utilization of the machine is always better using ATOptimal
- Response time is better than Toptimal but worse than HPC



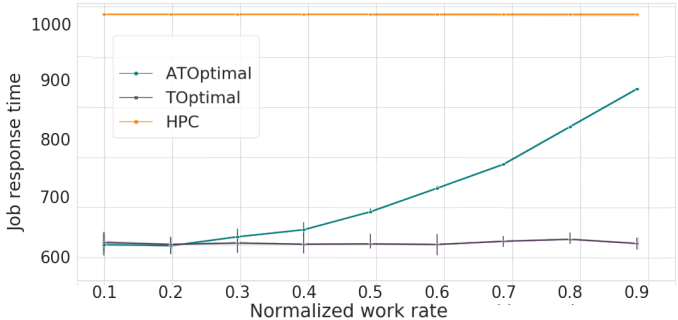
(c) Utilization



(d) Average job response time

Scenario 2: with known backfilling jobs

Average response time only for large jobs when varying the normalized work rate for backfilling jobs ζ



Scenario 3: Speculative backfilling

Backfill a job even if its reservation is larger than needed

- Choose the job that maximizes the expected utilization of the gap as follows
- In case the job fails it returns to its position in the waiting queue (no penalty)

For a gap of q processors and d duration:

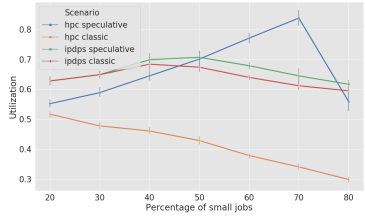
$$\max_{J_j \in \mathcal{J}'} G_j = \frac{p_j \int_{a'_j}^{d} t \cdot f'_j(t) dt}{q \cdot d}$$

a'_j and $f'_j(t) = f_j(t|t \geq a'_j)$ are the updated lower bound and PDF of the job

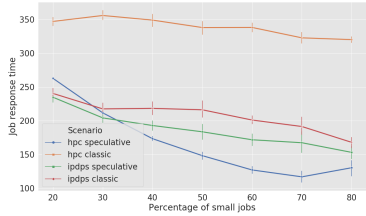
Scenario 3: Speculative backfilling

Varying the percentage of smaller jobs within the total number of jobs

- Small improvement for TOptimal compared to HPC
- Speculative HPC exceeds TOptimal for high number of small jobs



(e) Utilization



(f) Average job response time

Scenario 4: Simulating neuroscience on Intrepid

- Normalized rate of backfilling work ($\zeta = 0.21$)

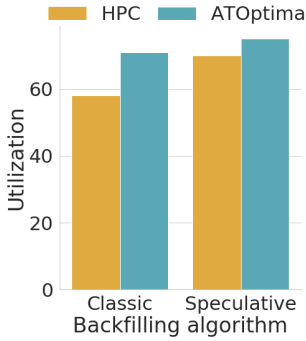
Application	Abdominal multi-organ segmentation
Distribution	Truncated Normal from 11 to 31 hours
Parameters	$\mu = 20$h and $\sigma = 8h$
# Submissions	10

Application	Whole brain segmentation and cortical reconstruction
Distribution	Truncated Normal from 1.5 to 3 hours
Parameters	$\mu = 1.7$h and $\sigma = 0.5$h
# Submissions	90

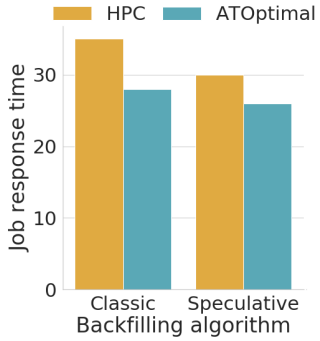
Application	FSL library of MRI and DTI analysis tools
Distribution	Truncated Normal from 10 to 35 minutes
Parameters	$\mu = 20$ min and $\sigma = 8$ min
# Submissions	300

Scenario 4: Simulating neuroscience on Intrepid

Simulating two weeks of neuroscience applications' execution on Intrepid



(g) Utilization



(h) Average job response time

Plan

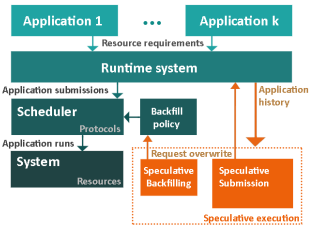
Conclusions

Pay what you use is not a viable solution for HPC system with the next generation of applications (or need lots of backfilling).

$\Longrightarrow$ Low system utilization, high response time.

We propose to introduce *Speculative Scheduling* on top of existing HPC schedulers.

- Job response time is decreased by 25%
- Overall effective utilization increases by 30%
- Processor idle time decreases, wasted computations increase (speculation)



Perspectives

Implementation issues:

- What can users provide to schedulers?
- Impact on power consumption?
- What is the overhead?

Single-app perspective (optim. of 1st phase):

- What if we can checkpoint the end of some/all reservations (coming up soon)
- How does this work with malleable jobs? (include more resources, nodes, memory)

Thanks