HPC-Big Data Convergence: Managing the Diversity of Application Profiles on HPC Facilities

Valentin Honoré
PhD defense

October 15, 2020
Under the supervision of Brice Goglin and Guillaume Pallez

High Performance Computing

Numerical simulations used in many scientific fields

- Biology,
- Physics,
- Cosmology,
- Chemestry,
- Industrial & Academics
- **Complexity of the problem, Massively Parallel**

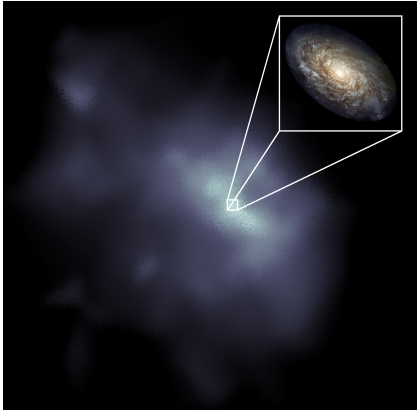


Figure: Dark matter halo image (Credit: ▸ CPAC group, ANL)

Supercomputers: large-scale machines

- Hierarchical layout
- Computing power: growth factor of 220 in last 10 years
- "Fugaku"
  - Equivalent to 1.5 million laptops
  - Energy consumption of $\sim$ 22.000 households

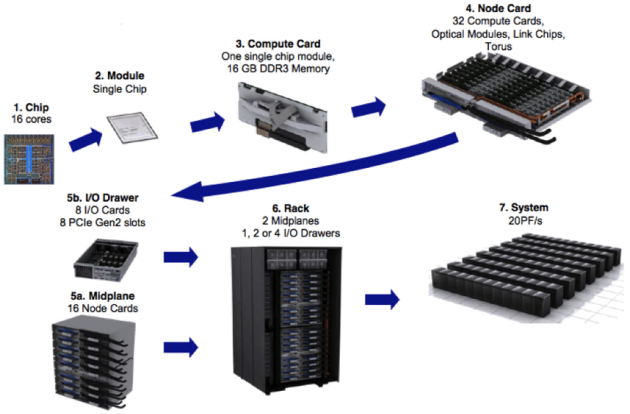


Figure: Structure of supercomputers: interconnected hierarchical boxes ▸ Source

Emergence of New Workloads

- Arise from Machine Learning and BigData
- Deep Learning as a major tool
  - Major interest in Artificial Intelligence
  - Neural Networks as the reference tool
- Progress: Larger datasets
- Consequence: compute-intensivity

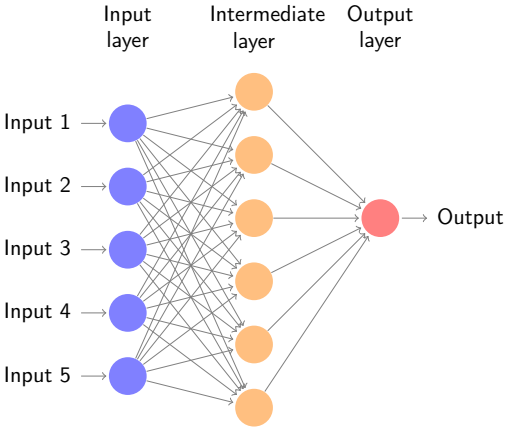
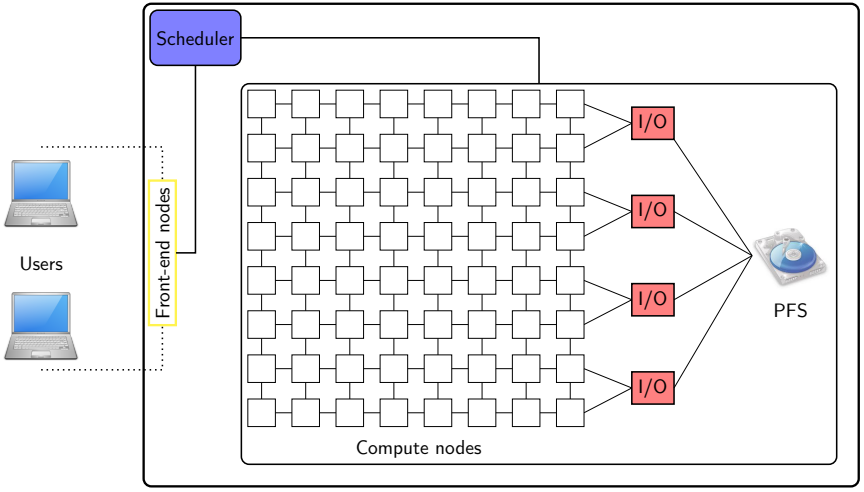


Figure: Illustration of neural network structure

Outline

Table of Contents

Running Applications on Supercomputers



Typical Scientific Applications

- Models of real-life phenomenon
- Complex code with high degree of parallelism
- *Output:* huge amount of data (datasets in PetaBytes)
- **Hardware Limitations:** limited size of PFS + access bandwidth
  - PFS capacity growth factor: 25 (2010-2020)*
  - Bandwidth growth factor: < 2 (2010-2020)*
- **Software Limitations:** Reaching perfect parallelism
  - Inherent sequential parts in applications
  - Amdahl's law, Gustafson law, etc

*Source: Top 500

Recently: new applications on supercomputers

- Convergence HPC/BigData-ML
- Our focus: the new applications
- Neurosciences, Computational Biology
- Applications: different models → different needs
- Machine: different users → different behavior

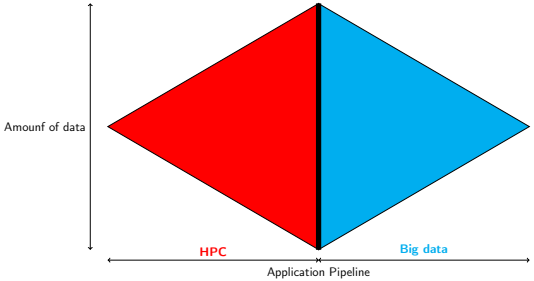


Figure: Convergence between HPC and BigData

Issue in Question

Issue in Question

How to optimize various application profiles on HPC infrastructures?

- Classify applications into categories
- Express models to derive efficient strategies

Issue in Question

Issue in Question

How to optimize various application profiles on HPC infrastructures?

- Classify applications into categories
- Express models to derive efficient strategies

	Data-intensive applications	**Stochastic Applications**
Origin	HPC workloads	New DL/HPC workloads
Challenge	Size of output	Important walltime variations
Approach	Scheduling, resource partitioning	Profiling, scheduling
Collaborations	B. Raffin (Inria)	Vanderbilt University, ENS Lyon
Publications	IJHPCA'19	IPDPS'19, IPDPS'20

Table: Overview of categories of applications under interest

Issue in Question

Issue in Question

How to optimize various application profiles on HPC infrastructures?

- Classify applications into categories
- Express models to derive efficient strategies

	Data-intensive applications	**Stochastic Applications**
Origin	HPC workloads	New DL/HPC workloads
Challenge	Size of output	Important walltime variations
Approach	Scheduling, resource partitioning	Profiling, scheduling
Collaborations	B. Raffin (Inria)	Vanderbilt University, ENS Lyon
Publications	IJHPCA'19	IPDPS'19, IPDPS'20

Table: Overview of categories of applications under interest

On the use of *in situ* paradigm (1/2)

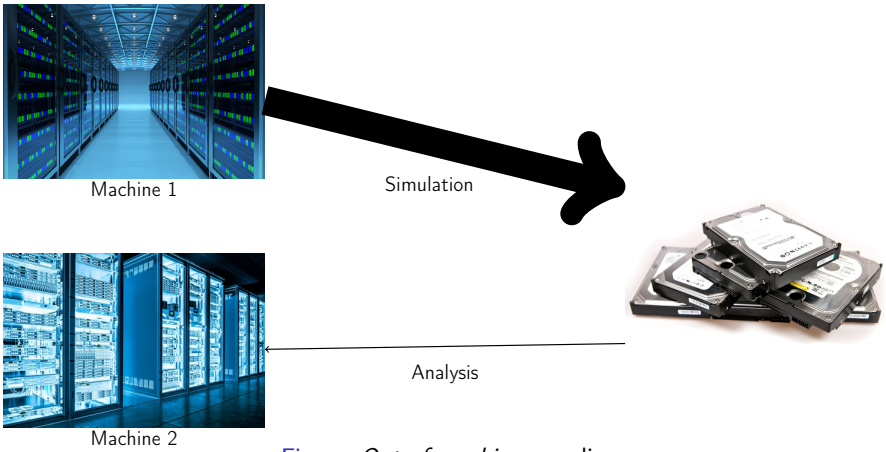


Figure: *Out-of-machine* paradigm

On the use of *in situ* paradigm (1/2)

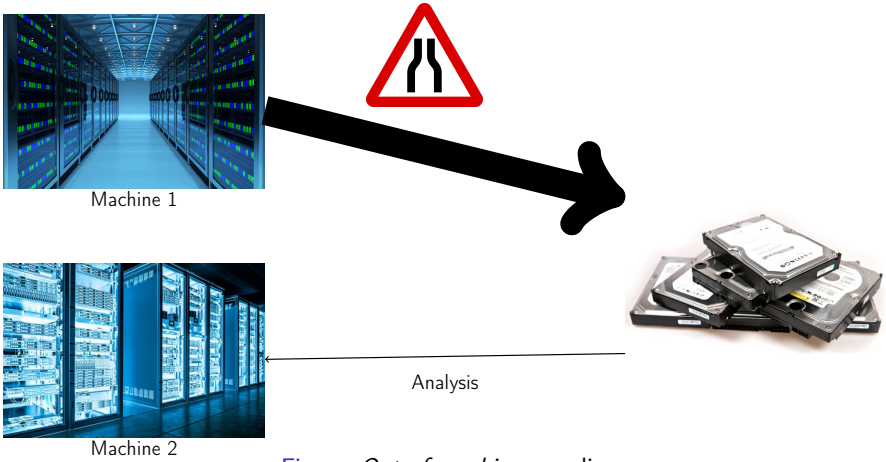


Figure: *Out-of-machine* paradigm

On the use of *in situ* paradigm (1/2)

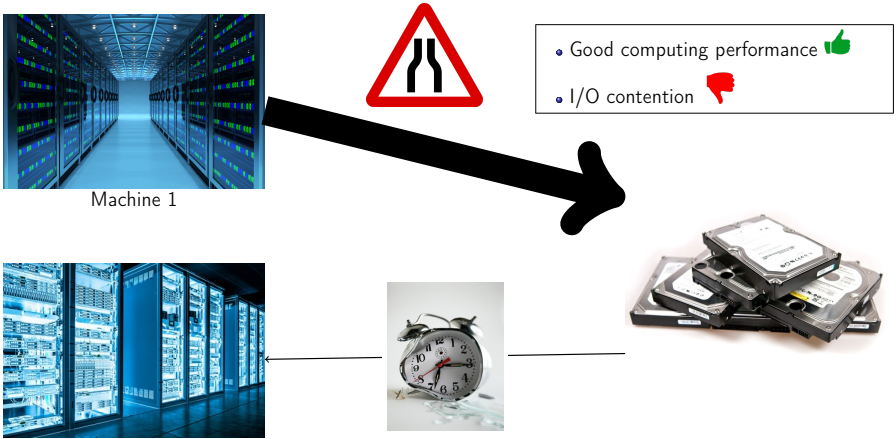


Figure: *Out-of-machine* paradigm

On the use of *in situ* paradigm (1/2)

Simulation

Analysis

Analysis output

- Reduce I/O contention
- Same machine, data locality
- Resources are shared between Sim. & Ana.
- **Need to optimize resources**

Figure: *In machine* paradigm

We propose models and strategies to automate *in machine* paradigm

On the use of *in situ* paradigm (2/2)

- Model of applications and platform
- Resource partitioning
- Scheduling of analysis
- Evaluation through simulations
- Perspectives: evaluation on production

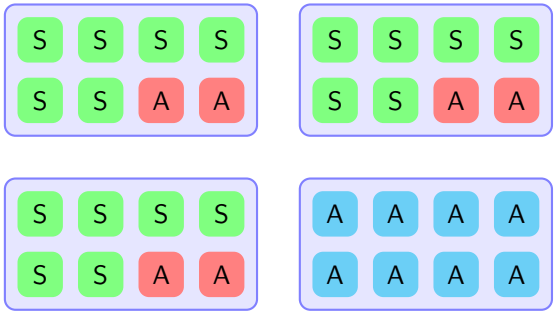


Figure: Hybrid *in transit*/*in situ* processing of analytics on 4 processors of 8 cores. Simulation runs on 6 cores and the *in situ* analytics on 2 helper cores of 3 *in situ* processors. *in transit* analytics are executed on a dedicated processor.

Issue in Question

Issue in Question

How to optimize various application profiles on HPC infrastructures?

- Classify applications into categories
- Express models to derive efficient strategies

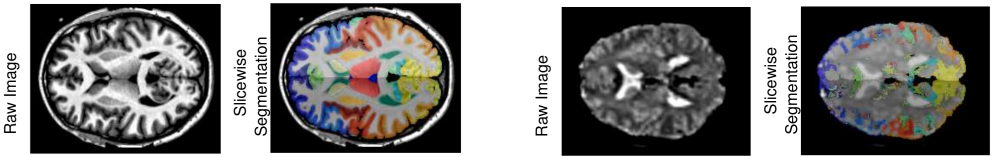
	Data-intensive applications	**Stochastic Applications**
Origin	HPC workloads	New DL/HPC workloads
Challenge	Size of output	Important walltime variations
Approach	Scheduling, resource partitioning	Profiling, scheduling
Collaborations	B. Raffin (Inria)	Vanderbilt University, ENS Lyon
Publications	IJHPCA'19	IPDPS'19, IPDPS'20

Table: Overview of categories of applications under interest

Table of Contents

SLANT, a representative Neuroscience application

- High-resolution whole brain segmentation from MRI pictures
- Dynamic code, multiple stages, predictable memory requirement
- Execution of CPU version, 2 datasets: DRD and OASIS
- Platform: Haswell on Plafrim



(a) Segmentation for OASIS.

(b) Segmentation for DRD.

Figure: Typical inputs and outputs based on the dataset.

High-level observations: walltime profile

- Variations are confirmed
- Where does these variations come?
- Can we predict it?
- Run-to-run variability? ✗

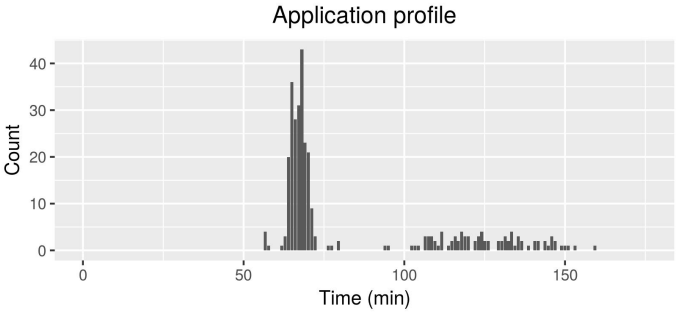


Figure: SLANT application walltime variation for various inputs.

High-level observations: walltime prediction?

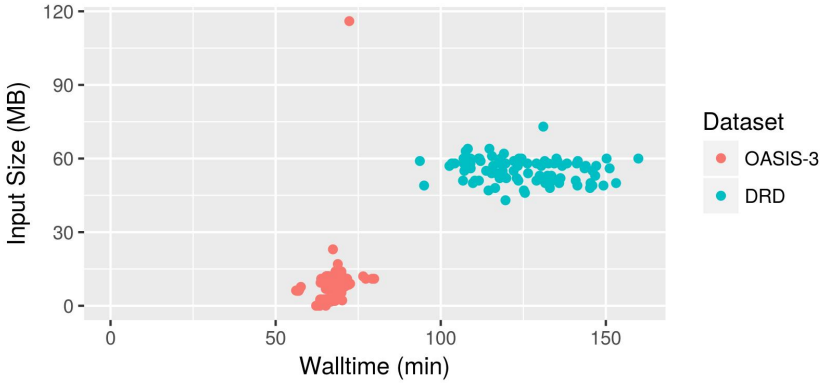


Figure: Correlation between the size of the input and the walltime over the 312 runs.

High-level observations: walltime prediction?

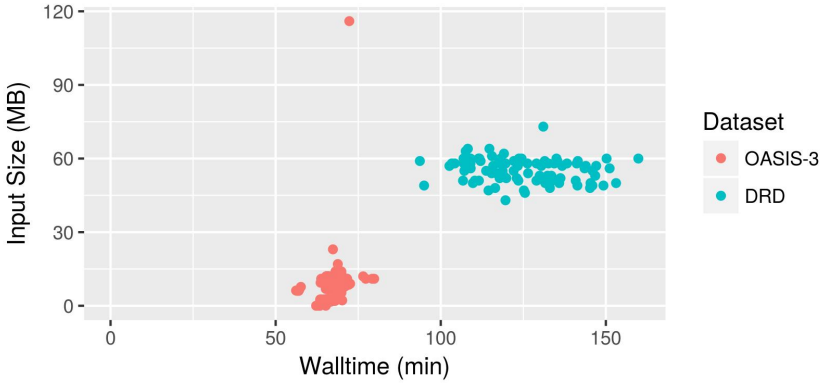
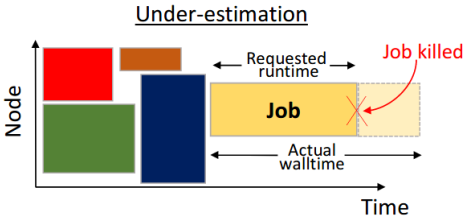


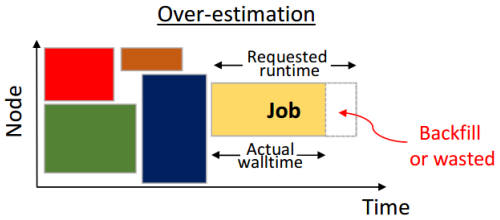
Figure: Correlation between the size of the input and the walltime over the 312 runs.

Let us accept this variation, in what is it a problem in HPC?

Difficulty of unknow execution time in HPC†



- Killed job + resubmission = low performance
- Waste of system resources



- Early termination, **possible waiting time in scheduler queue**
- Backfilling

Importance of accurate prediction of application walltime

†Credit pictures: Hongyang SUN

Overall high-level observations

- Significant variations in the walltime
- Variations determined by elements from the input
- Variations not correlated to the size of the input (quality and not quantity)
- **Approach: successive reservations until job terminates**

Reservation strategies for stochastic jobs

We provide

1. Model(s) of the applications
2. Model(s) of the platforms
3. An optimization problem.

Reservation-based Approach

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

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

Drawback: Failed reservations = Lost of the job progress = Waste of resources

Solution: introduce checkpoint at the end of (some) well-chosen reservations

- Assume flat memory model
- C: fixed checkpoint overhead
- R: restart overhead

Motivational example (1/2)

Job execution time follows a Random Variable X.

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

Let $X \sim \text{LogNormal}(21.4, 19.7)$ on $[0, 80]$

Platform

- Cost = cost of reservations
- $C = R = 7$

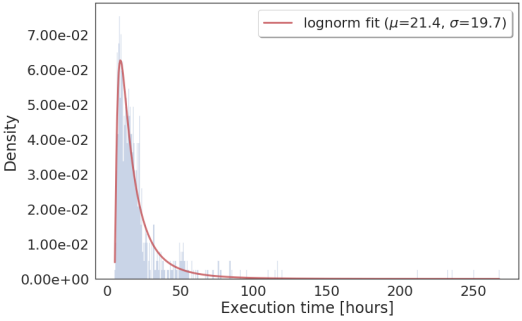
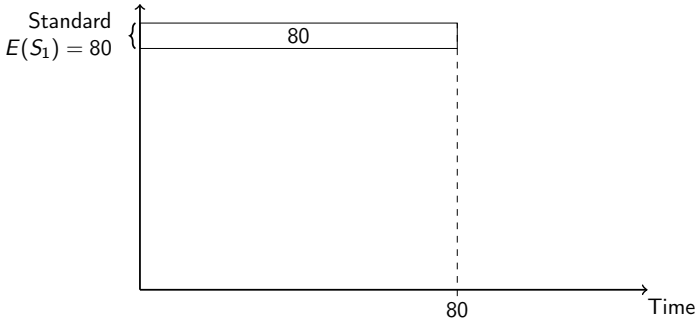


Figure: Execution times from 2017 for a *Structural identification of orbital anatomy* application, and its fitted distribution (in red).

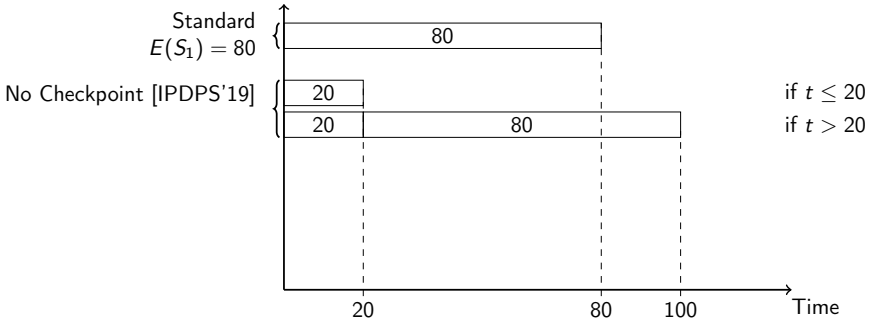
Checkpointing: Motivational example (2/2)

- S_1 (Standard): $t_1 = 80$



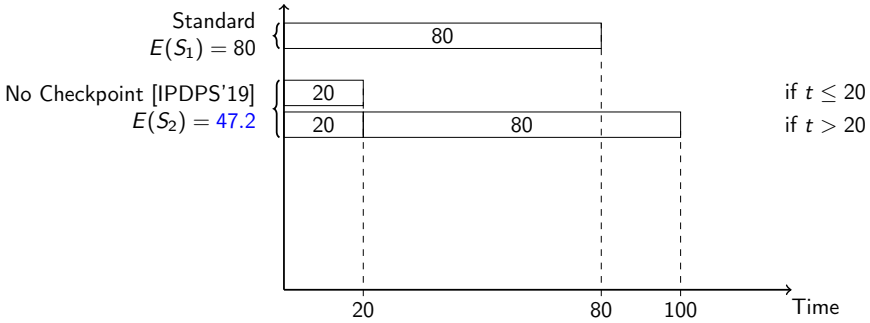
Checkpointing: Motivational example (2/2)

- S_2 (No Checkpoint): $t_1 = 20$, $t_2 = 80$



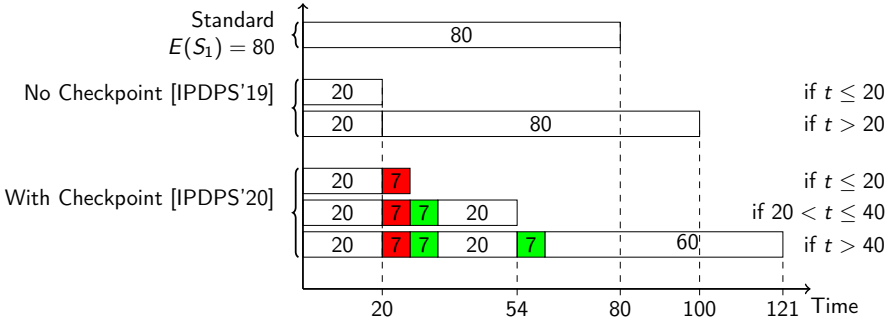
Checkpointing: Motivational example (2/2)

- $\mathbb{E}(S_2) = 20 \cdot \mathbb{P}(X \leq 20) + (80 + 20) \cdot \mathbb{P}(20 < X \leq 80) = 47.2$



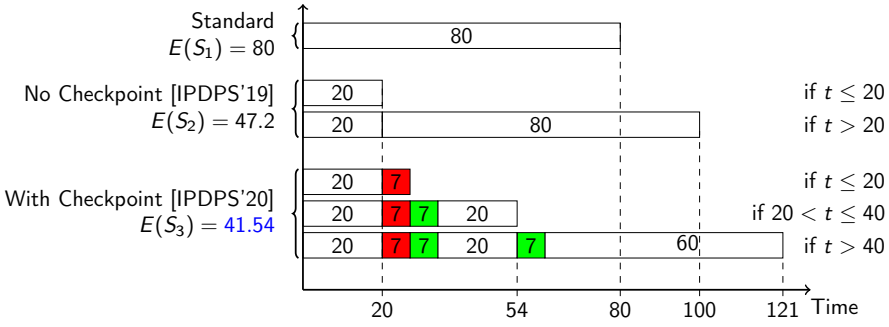
Checkpointing: Motivational example (2/2)

- S_3 (With Checkpoint): $t_1^C = 20 + C$, $t_2^R = R + 20$, $t_3^R = R + 60$



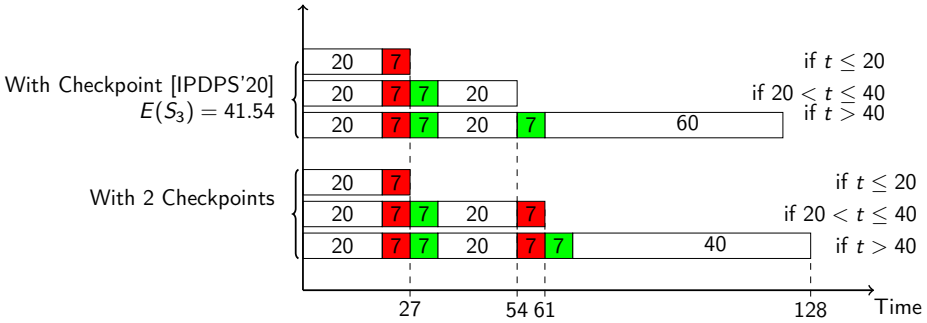
Checkpointing: Motivational example (2/2)

- $\mathbb{E}(S_3) = 27 \cdot \mathbb{P}(X \leq 20) + (27 + 27) \cdot \mathbb{P}(20 < X \leq 40) + (54 + 67) \cdot \mathbb{P}(40 < X) = 41.54$



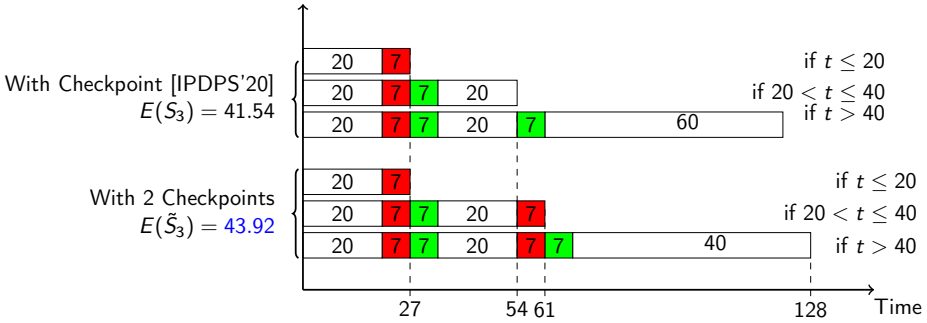
Checkpointing: Motivational example (2/2)

- $\tilde{S}_3$ (With 2 Checkpoints): $t_1^C = 20 + C$, $\tilde{t}_2^{R,C} = R + 20 + C$, $t_3^R = R + 60$



Checkpointing: Motivational example (2/2)

- $\mathbb{E}(\tilde{S}_3) = 27 \times 0.66 + 61 \times 0.26 + 128 \times 0.08 = 43.92 > \mathbb{E}(S_3)$



Complexity of checkpointing decision

Algorithmic contributions for stochastic jobs

- Checkpointing not always available
- Solutions for both application models

No checkpoints

- ⋆ Determine a sequence of reservations

With checkpoints:

- ⋆ Determine the size of reservations
- ⋆ Associate checkpointing decision to each reservation

Algorithmic contributions for stochastic jobs

No checkpoints

- ⋆ Determine a sequence of reservations

With checkpoints:

- ⋆ Determine the size of reservations
- ⋆ Associate checkpointing decision to each reservation

Distribution	Continuous		Discrete	
Support	Bounded	Unbounded	-	-
Solution	$(1+\epsilon)$-approx	?	optimal	optimal
Algorithm Name	Dyn-Prog-Count	Dyn-Prog-Count	Discrete-Ckpt	No-Checkpoint
Complexity	$\mathcal{O}\left(\frac{1}{\epsilon^3}\right)$	$\mathcal{O}\left(\frac{1}{\epsilon^3}\right)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$

Table: Summary of algorithmic contributions

Algorithmic contributions for stochastic jobs

No checkpoints

- ⋆ Determine a sequence of reservations

With checkpoints:

- ⋆ Determine the size of reservations
- ⋆ Associate checkpointing decision to each reservation

Name	All-Checkpoint	All-Checkpoint-Periodic	No-Checkpoint-Periodic
Distribution	Discrete	Discrete	Discrete
Application Model	Checkpoint	Checkpoint	No Checkpoint
Solution	Heuristic	Heuristic	Heuristic
Complexity	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

Table: Summary of main heuristics

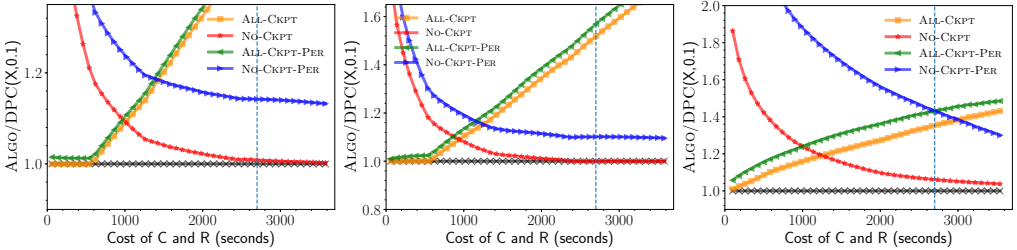
Simulations on synthetic applications

- Two sets of algorithms:
  1. "Checkpoint": Includes Dyn-Prog-Count [IPDPS'20], and its All-Checkpoint and No-Checkpoint variants [IPDPS'19]
  2. "Periodic": Includes All-Checkpoint-Periodic, and its No-Checkpoint-Periodic counterpart without checkpointing (i.e., $\delta_i = 0, \forall i$).

Simulations on synthetic applications

- Two sets of algorithms:
  1. "Checkpoint": Includes Dyn-Prog-Count [IPDPS'20], and its All-Checkpoint and No-Checkpoint variants [IPDPS'19]
  2. "Periodic": Includes All-Checkpoint-Periodic, and its No-Checkpoint-Periodic counterpart without checkpointing (i.e., $\delta_i = 0, \forall i$).
- Cost function
  1. "pay exactly what you reserve"
  2. Similar results for other cost functions
- Different distributions, bounded or not

Dyn-Prog-Count, best solution?



(a) Lognormal ($\mu = 45$min) (b) Bounded Pareto ($\mu = 45$min) (c) Weibull ($\mu = 45$min)

Figure: Expected costs of the different strategies normalized to that of Dyn-Prog-Count($X, 0.1$) when $C = R$ vary from 60 to 3600 seconds with ReservationOnly cost function.

Simulation: fit distribution from real data (1/2)

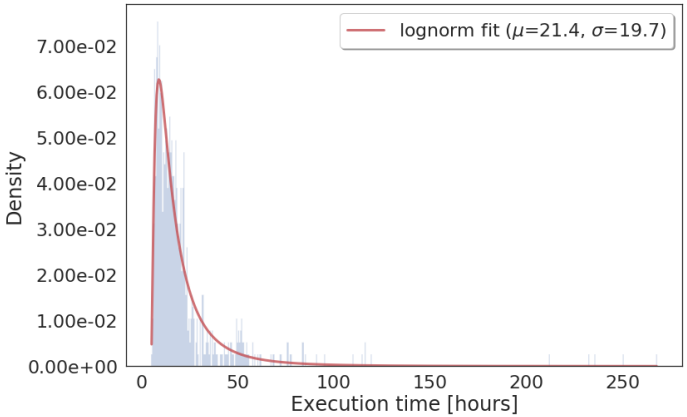
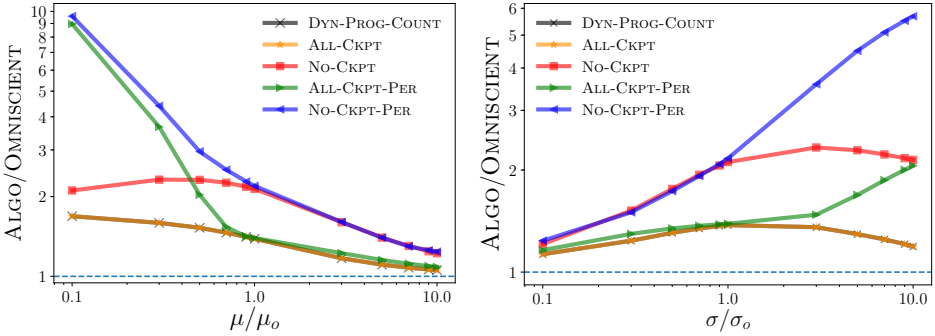


Figure: Execution times from 2017 for a *Structural identification of orbital anatomy* application, and its fitted distribution (in red) as a Lognormal distribution ($\mu_o = 21.4$h, $\sigma_0 = 19.7$h)

Simulation: fit distribution from real data (2/2)



(a) Variation of μ, $\sigma = \sigma_o = 19.7$h, $C = R = 600$s

(b) Variation of σ, $\mu = \mu_o = 21.4$h, $C = R = 600$s

Figure: Normalized performance of algorithms with omniscient scheduler when μ or σ vary, using ReservationOnly cost function.

Summary

- Reservation strategies for stochastic jobs
  - Exact and approximation algorithms for different application models
  - Set of simulation results to validate algorithm performance
  - First experiments in HPC environment
- Main limitations
  - Assume constant checkpoint/restart overhead
  - Inaccurate estimation leads to inefficiency

Summary

- Reservation strategies for stochastic jobs
  - Exact and approximation algorithms for different application models
  - Set of simulation results to validate algorithm performance
  - First experiments in HPC environment
- Main limitations
  - Assume constant checkpoint/restart overhead
  - Inaccurate estimation leads to inefficiency

Flat memory model

What if there are variations in memory consumption?
Is it the case in practice?

Table of Contents

Go back to SLANT

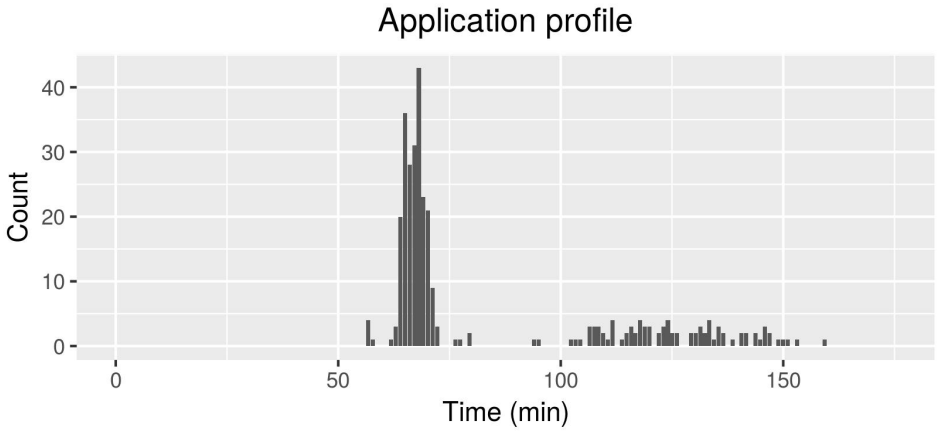


Figure: Reminder: SLANT execution time

Go back to SLANT

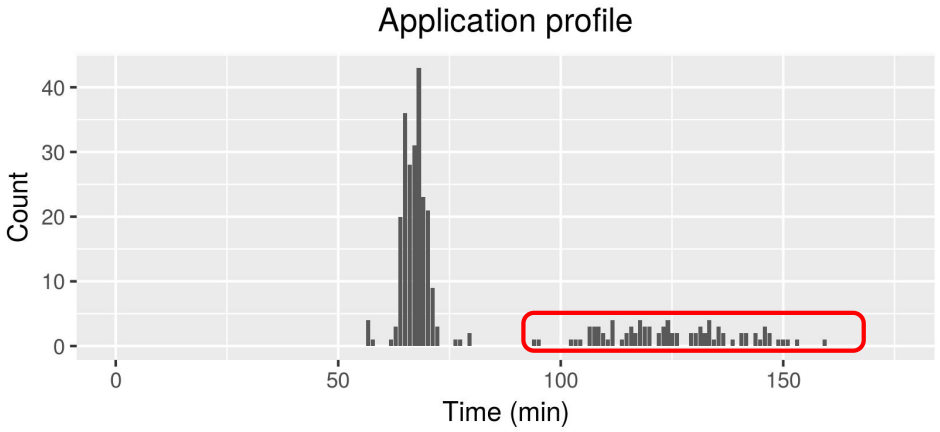


Figure: Reminder: SLANT execution time

SLANT: Task-level observations

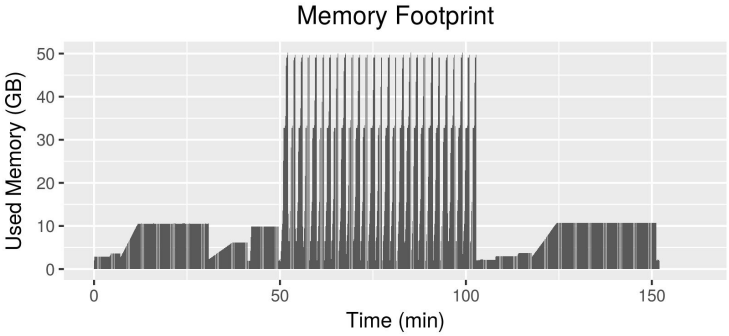


Figure: Examples of memory footprints of the SLANT application with inputs for DRD dataset. Memory consumption is measured every 2 seconds with the *used memory* field of the `vmstat` command.

Significant memory variations → Different checkpoint overhead

New application model: chain of tasks

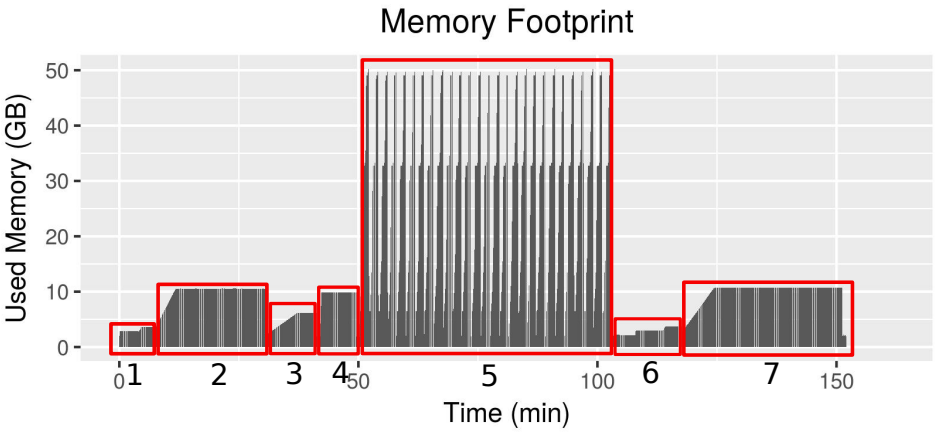


Figure: Job decomposition in tasks based on raw data of a memory footprint.

Task walltime: from raw data...

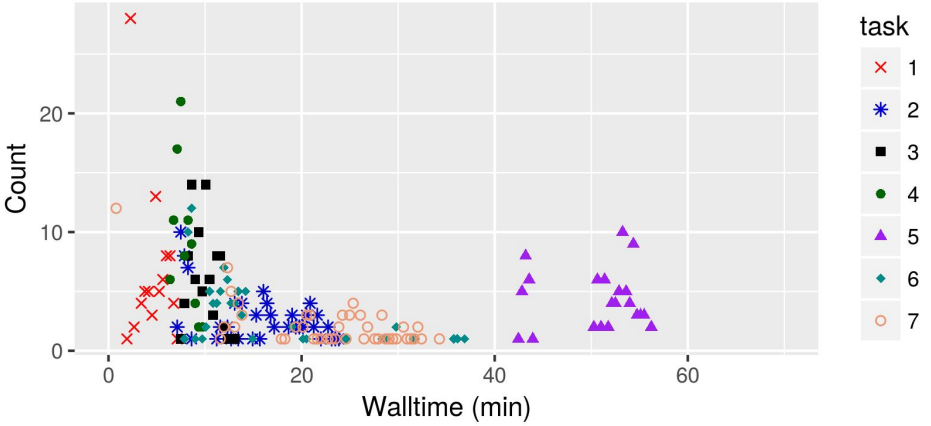


Figure: Analysis of the task walltime for all jobs (raw data).

... to model benefits (1/2)

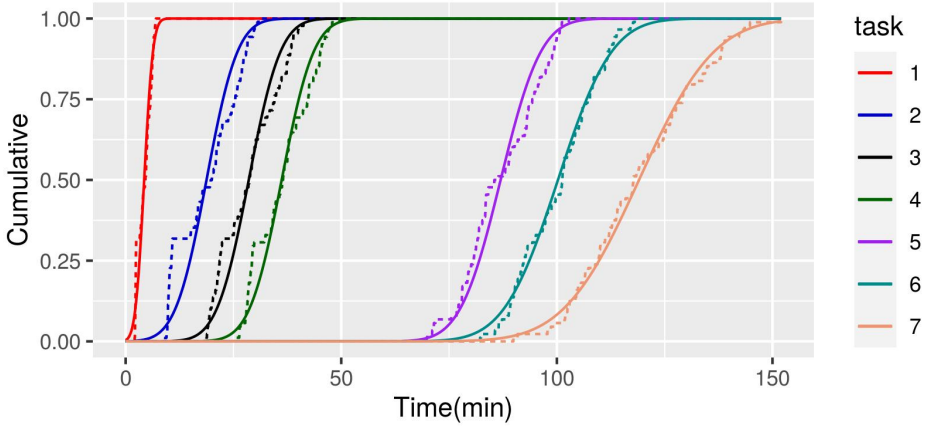


Figure: $y_i(t) = \mathbb{P}\left(\sum_{j \leq i} X_j < t\right)$ is the probability that task i is finished at time t (raw data and estimated).

Model benefits: Memory-specific quantities (2/2)

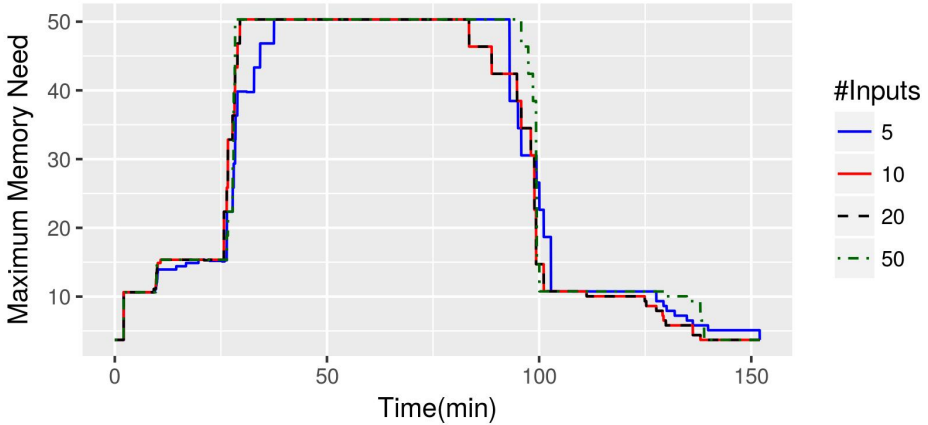


Figure: The model can help interpolate different quantities such as peak memory over time.

A first step for memory-aware algorithms

- Mem-All-Ckpt: adapting All-Checkpoint [IPDPS'20]
  1. Take k previous runs
  2. Interpolate model
     - estimate "likely" maximum memory need over time
     - From continuous to discrete distribution with associated checkpointing cost
  3. Input model to modified version of All-Checkpoint
- Comparison to strategies used at Vanderbilt University
  1. Neuro: T_1= max last 5runs, then T_2 s.t. $T_1 + T_2 = 1.5 \times T_1$
  2. Neuro-Avg: same with average of last 5runs

A first step for memory-aware algorithms

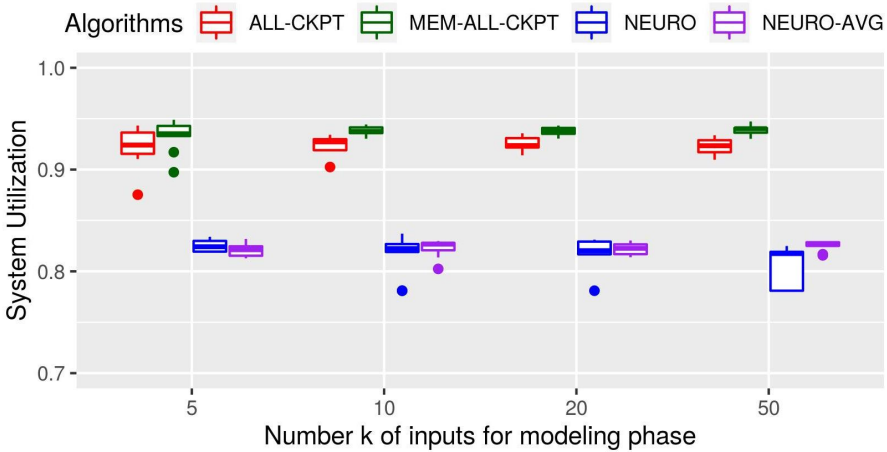


Figure: Average utilization (walltime divided by reservation time)

A first step for memory-aware algorithms

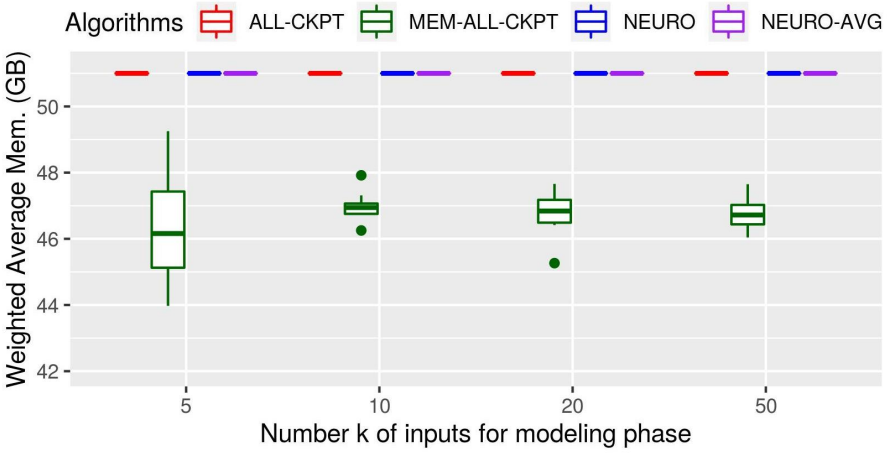


Figure: Weighted average memory (total memory used by the different reservations normalized by time)

Conclusion on stochastic applications

- Variation in execution time
- Sequence of reservations
  - Checkpoints to save application progress
  - Various algorithmic contributions
- Different performance evaluation
  - Simulations on synthetic applications
  - Experiments on real data and application
- Profiling applications for better strategies
  - Memory profiling to improve checkpointing strategy

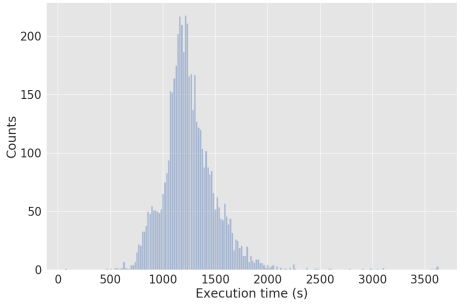


Figure: Traces [2013-2016] of VBMQA application (Vanderbilt's medical imaging database).

Table of Contents

Contributions to application management in HPC

- HPC/BigData-ML convergence
- Categorizing applications
- Derived strategies for each category
  - Expressed models for both platforms and applications
  - Derive scheduling heuristics and/or resource partitioning strategies
  - Extensive set of simulations and experiments
- Importance of application knowledge
  - Critical for performance of our strategies
  - Using profiling to better understand needs
  - From first solutions to practical ones

Short term perspectives

- Continue theoretical contributions
- Include system constraints in model
  - Contention
  - Checkpoint not only at the end of reservation
  - Application-aware checkpointing
- Integration in frameworks

Global perspectives

- Theoretical models, practical?
  - Convince users to use models
  - "Scheduler, where are you?"
- Study of users and application behavior at scale
- From static to dynamic decisions
  - Benefits of feedback during execution
  - Frequency of model revaluation
  - "dynamic" reservations

Thank you for your attention!

Context: Computing in the cloud

Several cost models to compute in the cloud:

- **On-Demand (OD):** "you pay for compute capacity by per hour or per second depending on which instances you run" (Amazon AWS).

(= Pay what you use)

- **Reserved-Instances (RI):** "Reserved Instances provide you with a significant discount (up to 75%) compared to On-Demand instance pricing."

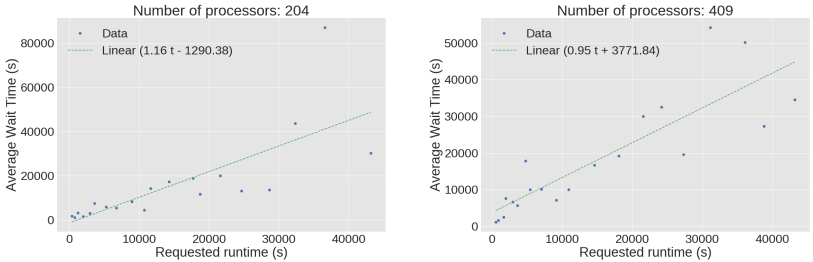
(= Pay what you ask for)

STANDARD 3-YEAR TERM					
Payment Option	Upfront	Monthly*	Effective Hourly**	Savings over On-Demand	On-Demand Hourly
No Upfront	$0	$8.03	$0.011	57%	$0.0255
Partial Upfront	$134	$3.72	$0.010	60%	
All Upfront	$252	$0.00	$0.010	62%	

Figure: Data extracted from AWS website, 12/10/2018.

Context: Computing in HPC

Total time = Wait time + Runtime:



(a) Jobs that requested 204 procs.

(b) Jobs that requested 409 procs.

Figure: Average wait times of jobs run on Intrepid (2009) as a function of requested runtime (data: Parallel Workload Archive).

Distribution Instantiation (1/2)

Distribution	PDF $f(t)$	Instantiation	Support (in hours)
Distributions with infinite support			
Exponential (λ)	$\lambda e^{-\lambda t}$	$\lambda = \frac{4}{3}h^{-1}$	$t \in [0, \infty)$
Weibull(λ, κ)	$\frac{\kappa}{\lambda}\left(\frac{t}{\lambda}\right)^{\kappa-1} e^{-\left(\frac{t}{\lambda}\right)^{\kappa}}$	$\lambda = 0.375h$ $\kappa = 0.5$	$t \in [0, \infty)$
Gamma(α, β)	$\frac{\beta^{\alpha}}{\Gamma(\alpha)} t^{\alpha-1} e^{-\beta t}$	$\alpha = 3.0$ $\beta = 4.0h^{-1}$	$t \in [0, \infty)$
Lognormal (ν, κ)	$\frac{1}{t\kappa\sqrt{2\pi}} e^{-\frac{(\ln t - \nu)^2}{2\kappa^2}}$	$\nu = 0.5h$ $\kappa = -0.789$	$t \in (0, \infty)$
Pareto(ν, α)	$\frac{\alpha \nu^{\alpha}}{t^{\alpha+1}}$	$\nu = 0.5h$ $\alpha = 3.0$	$t \in [\nu, \infty)$

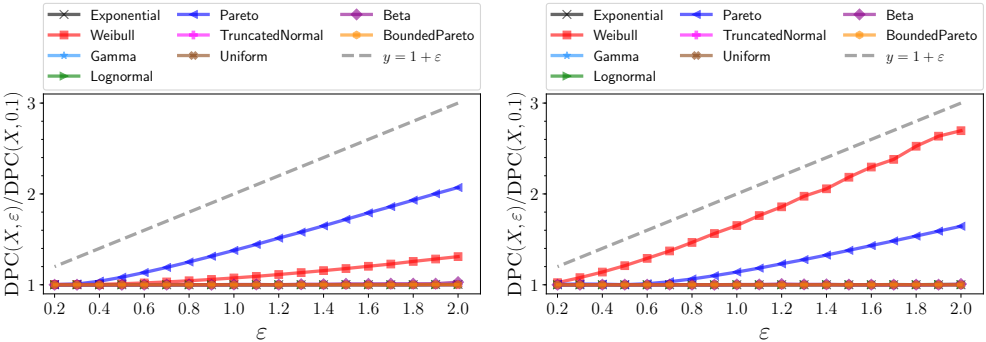
Table: Probability distributions and parameter instantiations.

Distribution Instantiation (2/2)

Distribution	PDF $f(t)$	Instantiation	Support (in hours)
Distributions with finite support			
Truncated Normal(ν, κ^2, a, b)	$\frac{1}{\kappa}\sqrt{\frac{2}{\pi}} \cdot \frac{e^{-\frac{1}{2}\left(\frac{t-\nu}{\kappa}\right)^2}}{1-\text{erf}\left(\frac{a-\nu}{\kappa\sqrt{2}}\right)}$	$\nu = 0.711h$ $\kappa^2 = 0.17h^2$ $a = 0.0h$ $b = 4.0h$	$t \in [a, b]$
Uniform(a, b)	$\frac{1}{b-a}$	$a = 0h$ $b = 1.5h$	$t \in [a, b]$
Beta(α, β)	$\frac{t^{\alpha-1}\cdot(1-t)^{\beta-1}}{\text{B}(\alpha,\beta)}$	$\alpha = 2.0$ $\beta = \frac{2}{3}$	$t \in [0, 1]$
Bounded Pareto(L, H, α)	$\frac{\alpha L^\alpha t^{-\alpha-1}}{1-\left(\frac{L}{H}\right)^\alpha}$	$L = 0.233h$ $H = 5.0h$ $\alpha = 1.0$	$t \in [L, H]$

Table: Probability distributions and parameter instantiations.

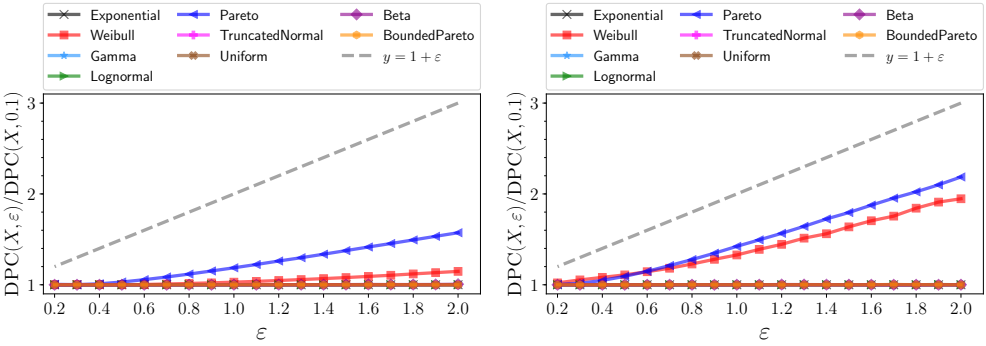
Impact of ε on Dyn-Prog-Count



(a) $C = R = 6\text{min}$ (b) $C = R = 30\text{min}$

Figure: Expected cost of Dyn-Prog-Count(X, ε) as a function of ε for different distributions with ReservationOnly cost function.

Impact of ε on Dyn-Prog-Count



(a) $C = R = 6\text{min}$ (b) $C = R = 30\text{min}$

Figure: Expected cost of Dyn-Prog-Count(X, ε) as a function of ε for different distributions with HPC cost function.

Impact of period size (1/2)

Table: Expected cost of All-Checkpoint-Periodic and No-Checkpoint-Periodic, normalized by Dyn-Prog-Count($X, 0.1$) for $C = R = 360$s, with ReservationOnly cost function.

Distribution	All-Checkpoint-Periodic							No-Checkpoint-Periodic						
	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$
Exponential	25 (1.00)	9.03	2.44	4.28	6.13	7.98	9.84	13 (1.38)	9.03	7.53	14.48	21.43	28.38	35.33
Weibull	223 (1.12)	76.76	1.13	1.22	1.42	1.64	1.86	69 (2.35)	76.76	3.56	6.16	8.85	11.56	14.28
Gamma	13 (1.02)	5.09	3.66	6.69	9.71	12.74	15.77	7 (1.33)	5.09	11.99	23.35	34.71	46.07	57.43
Lognormal	20 (1.01)	7.71	2.90	5.19	7.48	9.78	12.08	10 (1.52)	7.71	11.52	22.44	33.37	44.30	55.23
Pareto	429 (1.01)	96.28	1.13	1.01	1.03	1.08	1.15	267 (1.16)	96.28	1.19	1.22	1.37	1.55	1.75
TruncatedNormal	7 (1.06)	3.25	6.72	12.81	18.91	25.01	31.10	4 (1.23)	3.25	14.91	29.21	43.51	57.80	72.10
Uniform	3 (1.01)	1.20	16.59	32.57	48.56	64.54	80.53	1 (1.20)	1.20	40.56	80.53	120.49	160.45	200.41
Beta	1 (1.00)	1.00	30.75	60.75	90.75	120.75	150.75	1 (1.00)	1.00	62.11	123.47	184.84	246.20	307.57
BoundedPareto	13 (1.02)	3.99	4.06	7.52	10.97	14.42	17.88	6 (1.44)	3.99	15.17	29.73	44.29	58.85	73.41

Impact of period size (2/2)

Table: Expected cost of All-Checkpoint-Periodic and No-Checkpoint-Periodic, normalized by Dyn-Prog-Count($X, 0.1$) for $C = R = 1800$s, with ReservationOnly cost function.

Distribution	All-Checkpoint-Periodic							No-Checkpoint-Periodic						
	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$	Best τ	$\tau = 1$	$\tau = 200$	$\tau = 400$	$\tau = 600$	$\tau = 800$	$\tau = 1000$
Exponential	13 (1.26)	6.86	7.49	14.52	21.56	28.61	35.65	13 (1.05)	6.86	5.72	11.00	16.28	21.56	26.84
Weibull	108 (1.34)	52.64	1.52	2.23	3.02	3.83	4.65	69 (1.61)	52.64	2.44	4.23	6.07	7.93	9.79
Gamma	7 (1.17)	3.83	11.86	23.24	34.63	46.01	57.40	7 (1.00)	3.83	9.01	17.55	26.09	34.63	43.17
Lognormal	11 (1.29)	5.91	9.27	18.08	26.89	35.70	44.52	10 (1.17)	5.91	8.83	17.20	25.57	33.95	42.33
Pareto	228 (1.39)	83.16	1.40	1.52	1.81	2.13	2.47	267 (1.01)	83.16	1.03	1.05	1.18	1.34	1.51
TruncatedNormal	3 (1.40)	2.81	26.83	53.13	79.43	105.73	132.03	4 (1.06)	2.81	12.86	25.19	37.52	49.86	62.19
Uniform	1 (1.00)	1.00	67.17	133.83	200.50	267.17	333.83	1 (1.00)	1.00	33.83	67.17	100.50	133.83	167.17
Beta	1 (1.00)	1.00	150.72	300.73	450.74	600.74	750.74	1 (1.00)	1.00	62.11	123.47	184.84	246.20	307.57
BoundedPareto	7 (1.33)	3.10	13.92	27.35	40.79	54.22	67.66	6 (1.12)	3.10	11.80	23.12	34.44	45.77	57.10

From simulation to experiments

- Scheduling multiple jobs in an HPC environment

From simulation to experiments

- Scheduling multiple jobs in an HPC environment
- 3 different applications

Table: Characteristics of the chosen neuroscience applications.

Application Type	**Walltime distribution**	C	R
Diffusion model fitting (Qball)	Gamma ($k = 1.18$, $\theta = 34$, $[a, b] = [146s, 407s]$)	90s	40s
Diffusion model fitting (SD)	Weibull ($k = 1043811$, $\lambda = 1174322466$, $[a, b] = [46min, 2.3h]$)	25min	10min
Functional connectivity analysis (FCA)	Gamma ($k = 3.6$, $\theta = 72$, $[a, b] = [165s, 1003s]$)	150s	100s

From simulation to experiments

- Scheduling multiple jobs in an HPC environment
- 3 different applications
- Two different sets of strategies
  1. "HPC-for-neuroscience strategy" (HPC): average of 5 last runs then $\times 1.5$
  2. Our Dyn-Prog-Count strategy + All-Checkpoint variant

From simulation to experiments

- Setup
  - 256-thread Intel Processor Xeon Phi 7230, 1.30GHz
  - Record application completion time, for 500 runs of each application
  - Metrics for system performance:
    1. stretch: total execution time / actual walltime
    2. utilization: sum of all jobs' walltimes / total time required to execute them

Experiments: performance of strategies

- 500 jobs of each applications are submitted
- C/R manually forced to be as presented in previous table (study of application interference and runtime variability)

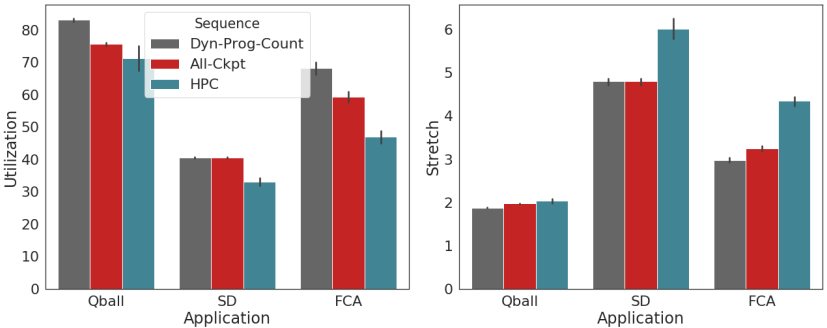


Figure: Utilization and average job stretch for Dyn-Prog-Count, All-Checkpoint and the HPC strategies.

Experiments: what if C/R vary?

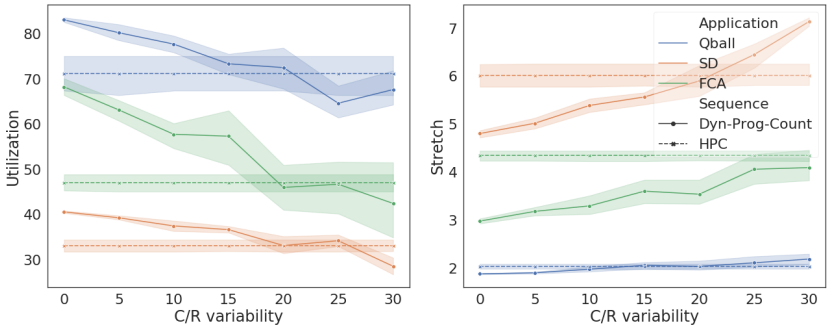


Figure: Utilization and average job stretch for the three applications (blue: Qball; Orange: SD; Green: FCA) when varying the C/R costs by different percentages (0 to 30%) using the Dyn-Prog-Count strategy. Horizontal lines represent the results for the HPC strategy.

Experiments: what about application interference?

- Running all three applications at same time
- 500 jobs (100 from Qball, and 200 each from SD and FCA)
- C/R costs constant across different reservations
- HPC strategy: 10 different runs choosing 10 different instances from traces

Experiments: what about application interference?

Table: Utilization and average job stretch for 10 different runs, each using 500 jobs from all three applications. The runs are ordered by the best improvement of Dyn-Prog-Count in utilization.

Dyn-Prog-Count		HPC		Improvement	
Utilization	**Avg Stretch**	**Utilization**	**Avg Stretch**	**Utilization**	**Avg Stretch**
67	2.04	55	2.34	21%	15%
73	1.72	62	2.04	18%	19%
62	2.08	55	2.46	12%	18%
71	1.88	64	2.1	11%	12%
63	2.19	56	2.41	11%	10%
71	1.74	64	1.96	10%	12%
75	1.51	68	1.69	10%	12%
68	2.09	65	2.19	4%	5%
61	2.24	60	2.32	2%	4%
77	1.96	75	1.99	2%	2%

Mem-All-Ckpt heuristic

- "likely" maximum memory needed as a function of time

$$M_\tau(t) = \max\left\{ M_i | \mathbb{P}\left(\sum_{j<i} X_i < t \leq \sum_{j \leq i} X_i \right) > \tau \right\} \quad (1)$$

- $Y \sim (v_i, C_i, f_i)_{1 \leq i \leq n}$, s.t. for $1 \leq i \leq n$, $\mathbb{P}(Y = v_i) = f_i$
- Dynamic-programming algorithm

$$\begin{cases} S_{\text{MAC}}(n) = 0 \\ S_{\text{MAC}}(i) = \min\limits_{i+1 \leq j \leq n} \left(S_{\text{MAC}}(j) + (R + (v_j - v_i) + C_i) \cdot \sum\limits_{k=i+1}^{n} f_k \right) \end{cases}$$

- Checkpoint cost C_i
  - All-Checkpoint: Max. memory peak of application
  - Mem-All-Ckpt: $M_\tau(v_i)$

A first step for memory-aware algorithms

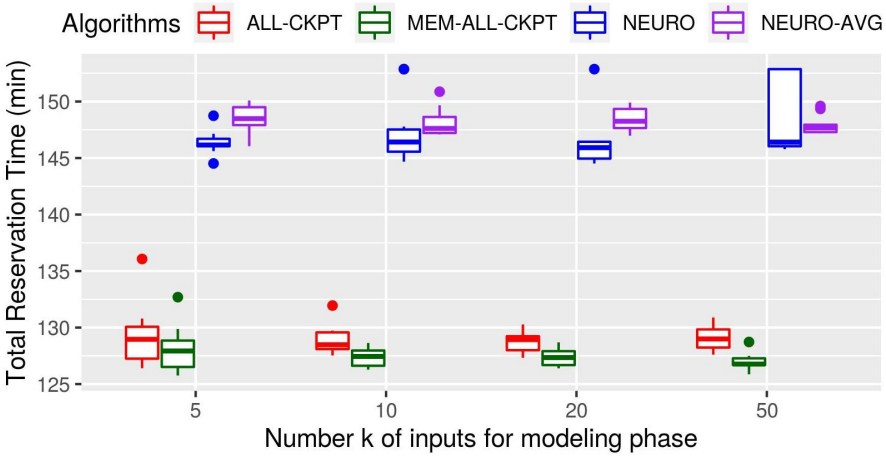


Figure: Average reservation time

A "second" step for memory-aware algorithms

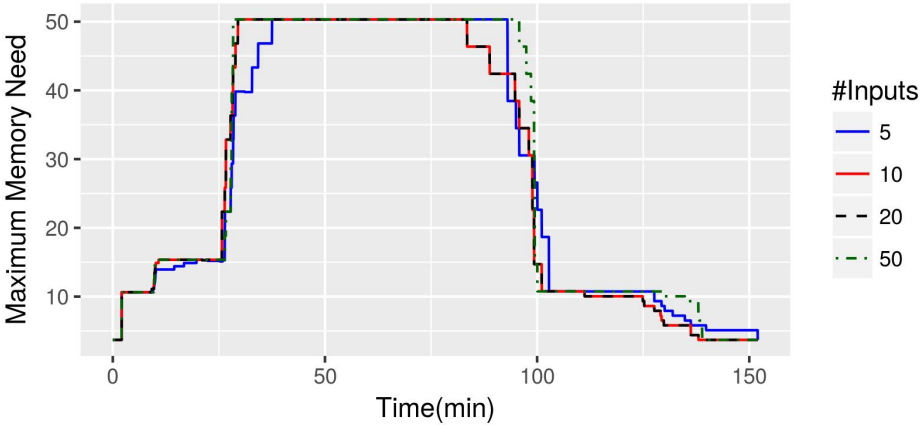


Figure: Interpolation of peak memory over time.

A "second" step for memory-aware algorithms

- Mem-All-Ckpt: first reservation of >100min
- R_1: memory peak of 50GB
- Enhancement: checkpoint just before task 5 (<25min)
- Mem-All-Ckptv2

A "second" step for memory-aware algorithms

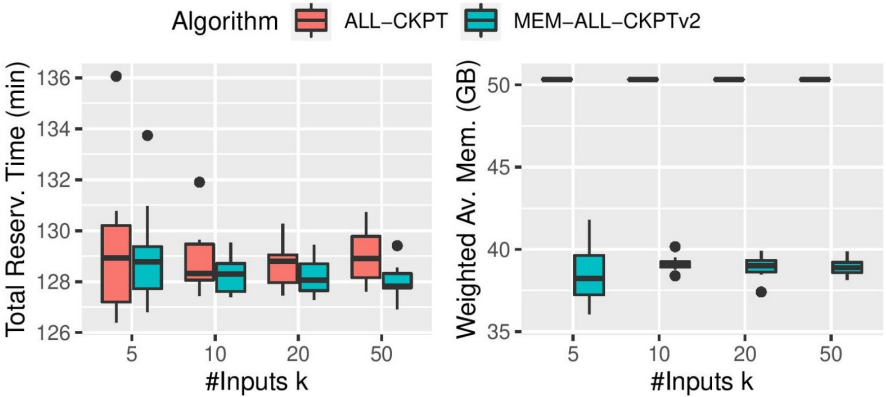


Figure: Average reservation time

Usual speedup laws

Amdahl's law

$$S = \frac{1}{(1-p) + \frac{p}{s}} \quad (2)$$

- S = speedup
- p = fraction of code that benefits
- s = speedup of fraction p

Difference: Amdahl assumes fixed problem size

Gustafson's law

$$S = N + (1-N) \times s \quad (3)$$

- S = speedup
- N = processors
- s = sequential fraction

Covid-friendly buffet!!!