Reducing Costs of Spot Instances via Checkpointing in the Amazon Elastic Compute Cloud

Sangho Yi, Derrick Kondo, and Artur Andrzejak

- at CLOUD 2010 -

Sangho Yi
INRIA Grenoble, France
(sangho.yi@inria.fr)

Outline

- Motivation and background
  - Spot Instances on Amazon EC2
- Possible checkpointing strategies
- Performance evaluation
- Conclusions and future work

Motivation:
Market-based Pricing in Cloud

- Amazon released 'Spot instances' in Dec. 2009.
  - Current price depends on users' bids, and available computing resources.
    - (based on the '**supply** and **demand**' relationship)
  - 30~50% price of normal instances.
  - In-bid users get instances.
  - Out-of-bid users lose their instances.

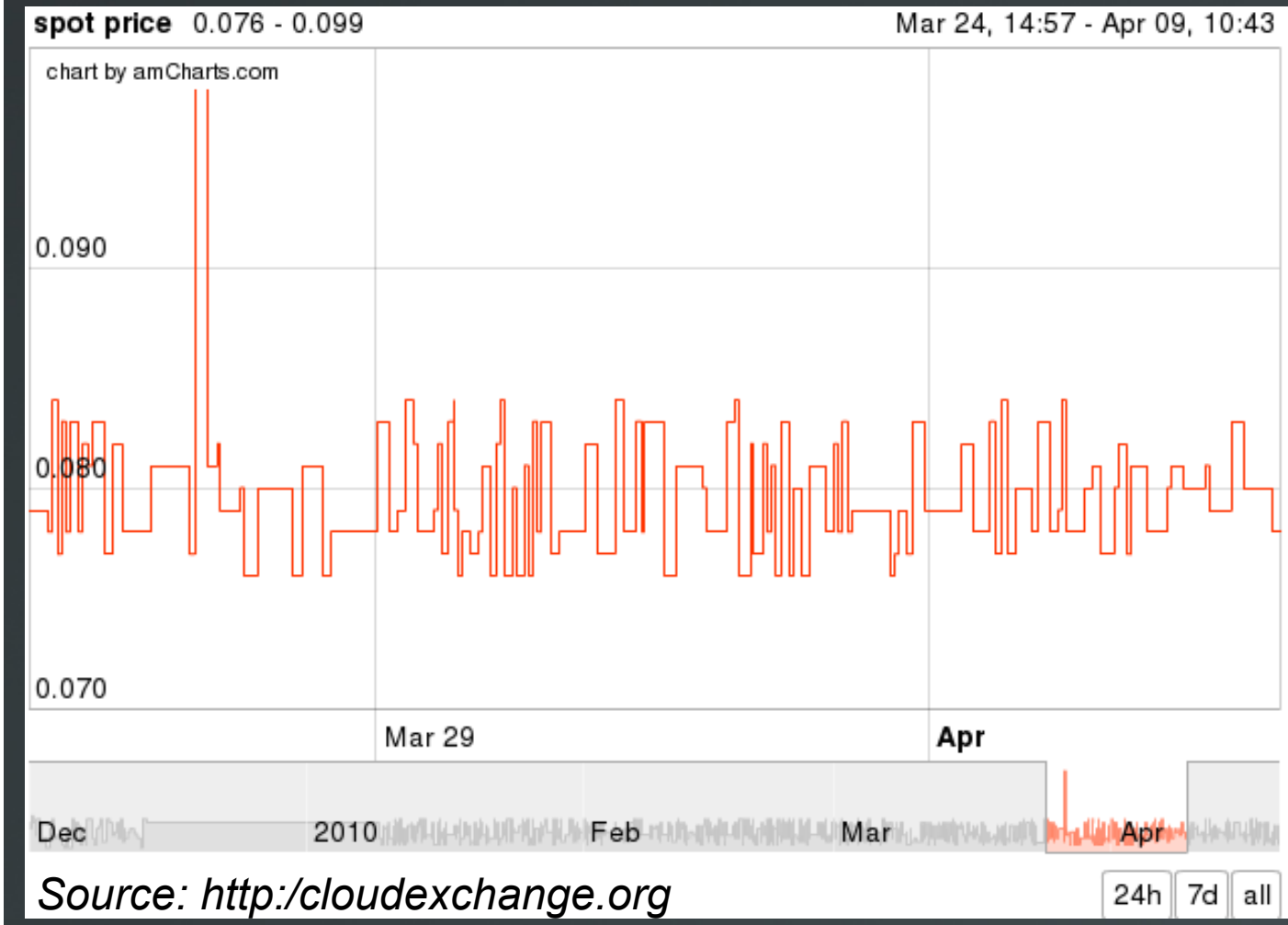
Pricing in Amazon EC2 (normal instances – on demand)

Virtual Machine Instance Type	Linux Cost/hour (USD)	MS Windows Cost/hour (USD)
Standard Small	0.095	0.12
Standard Large	0.38	0.48
Standard Extra Large	0.76	0.96
High CPU - Medium	0.19	0.29
High CPU - Extra Large	0.76	1.16

** Pricing on eu-west (Ireland), 2010

Data Transfer Type	Cost/GB-Month (USD)
Inbound Transfer	0.10
First 10 TB	0.17
Next 10-50 TB	0.13
Next 50-150 TB	0.11
Over 150 TB	0.10

Pricing in Amazon EC2 (spot-instances – bidding system)

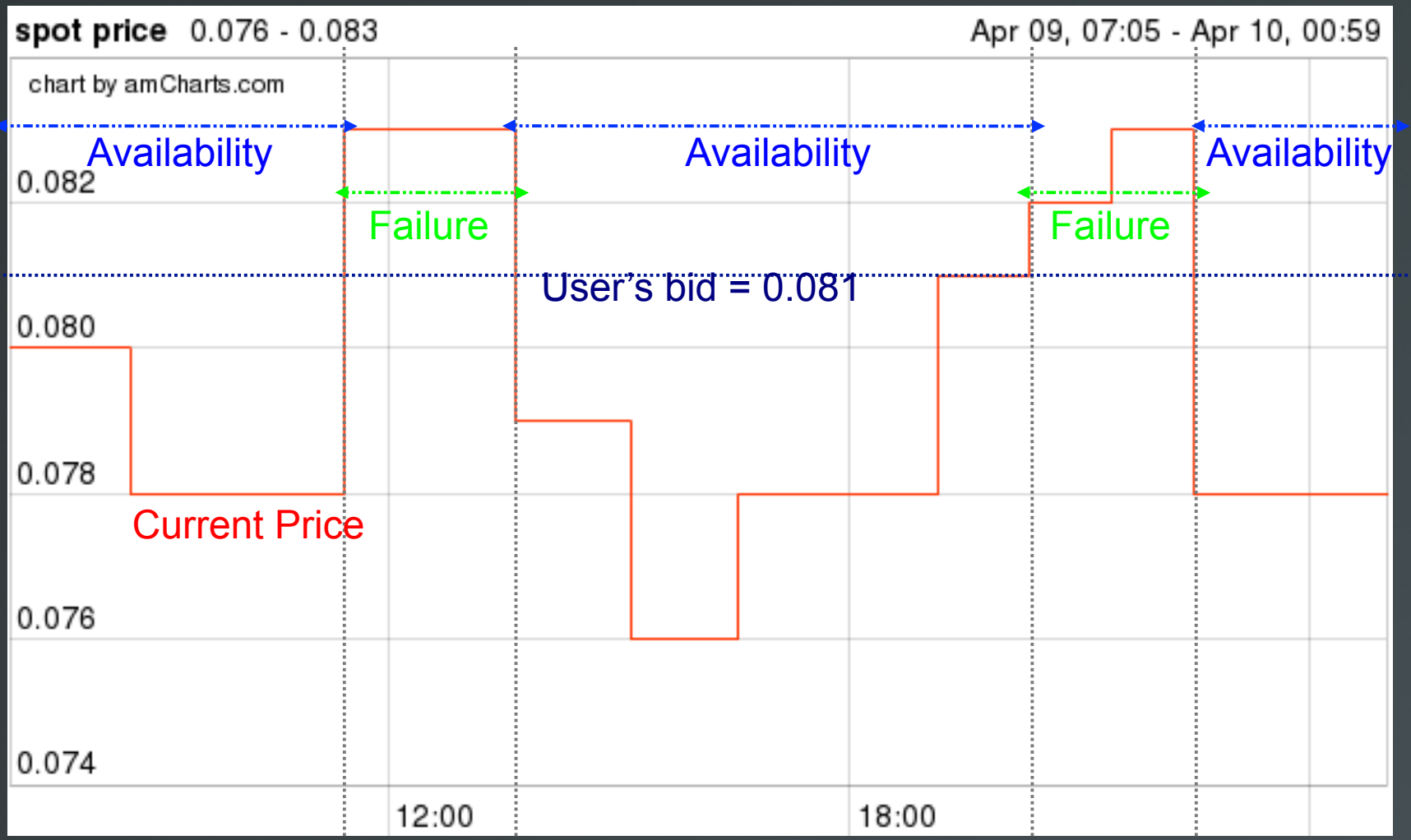


linux	us-west-1	us-east-1	eu-west-1
m1.small	0.038 | 40	0.030 | 35	0.039 | 41
m1.large	0.154 | 41	0.122 | 36	0.167 | 44
m1.xlarge	0.335 | 44	0.241 | 35	0.311 | 41
c1.medium	0.076 | 40	0.062 | 36	0.077 | 41
c1.xlarge	0.313 | 41	0.245 | 36	0.314 | 41
m2.xlarge	0.248 | 44	0.200 | 40	0.240 | 42
m2.2xlarge	0.563 | 42	0.425 | 35	0.572 | 43
m2.4xlarge	1.073 | 40	0.821 | 34	1.152 | 43

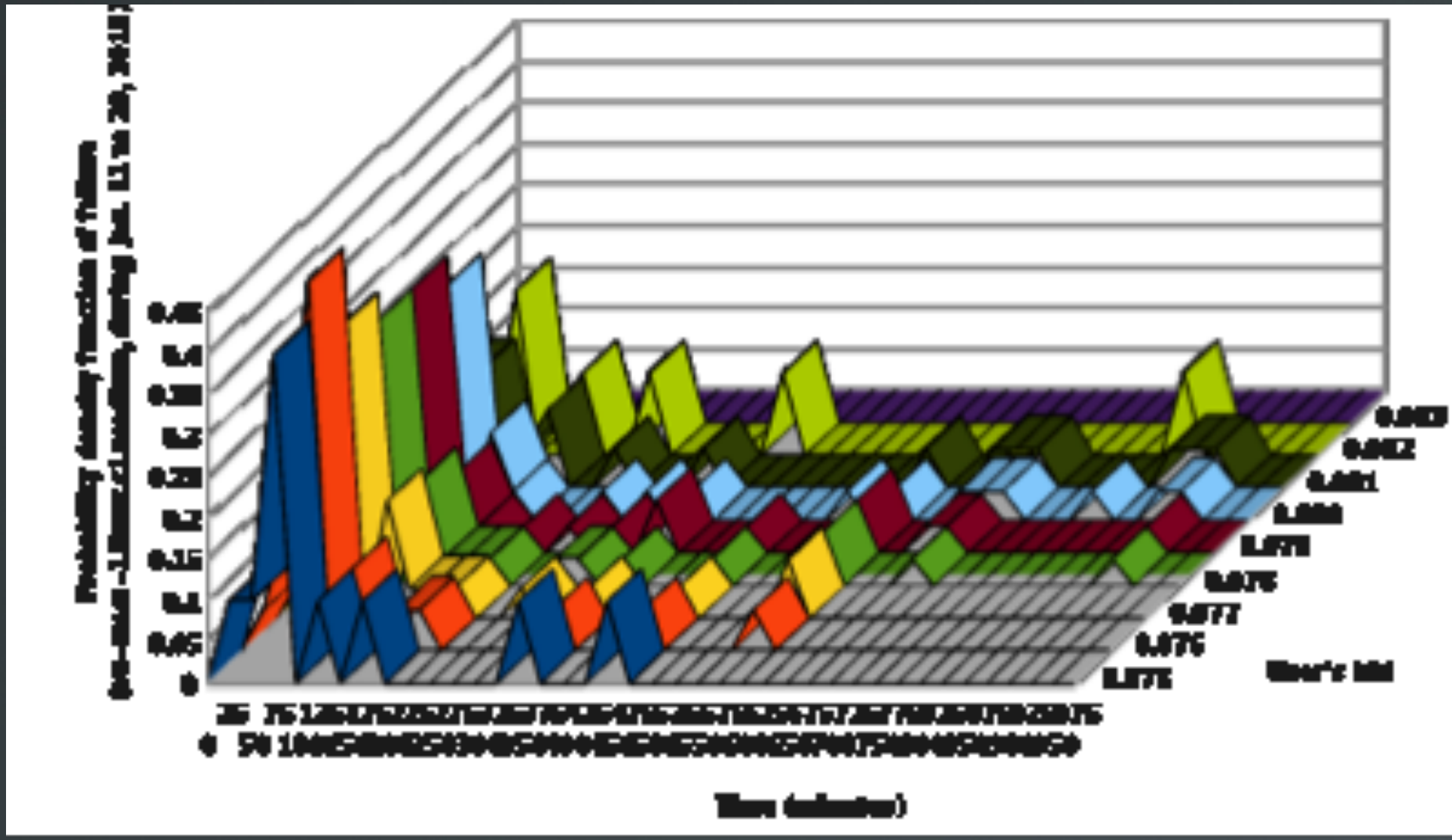
windows	us-west-1	us-east-1	eu-west-1
m1.small	0.069 | 53	0.052 | 43	0.067 | 56
m1.large	0.254 | 49	0.203 | 42	0.273 | 57
m1.xlarge	0.517 | 50	0.387 | 40	0.532 | 55
c1.medium	0.160 | 52	0.127 | 44	0.169 | 58
c1.xlarge	0.694 | 56	0.476 | 41	0.657 | 57
m2.xlarge	0.305 | 44	0.235 | 38	0.326 | 53
m2.2xlarge	0.710 | 45	0.549 | 38	0.720 | 50
m2.4xlarge	1.421 | 45	1.146 | 40	1.492 | 52

Source: http:/cloudexchange.org

An Example of Failures as a Function of User's Bid

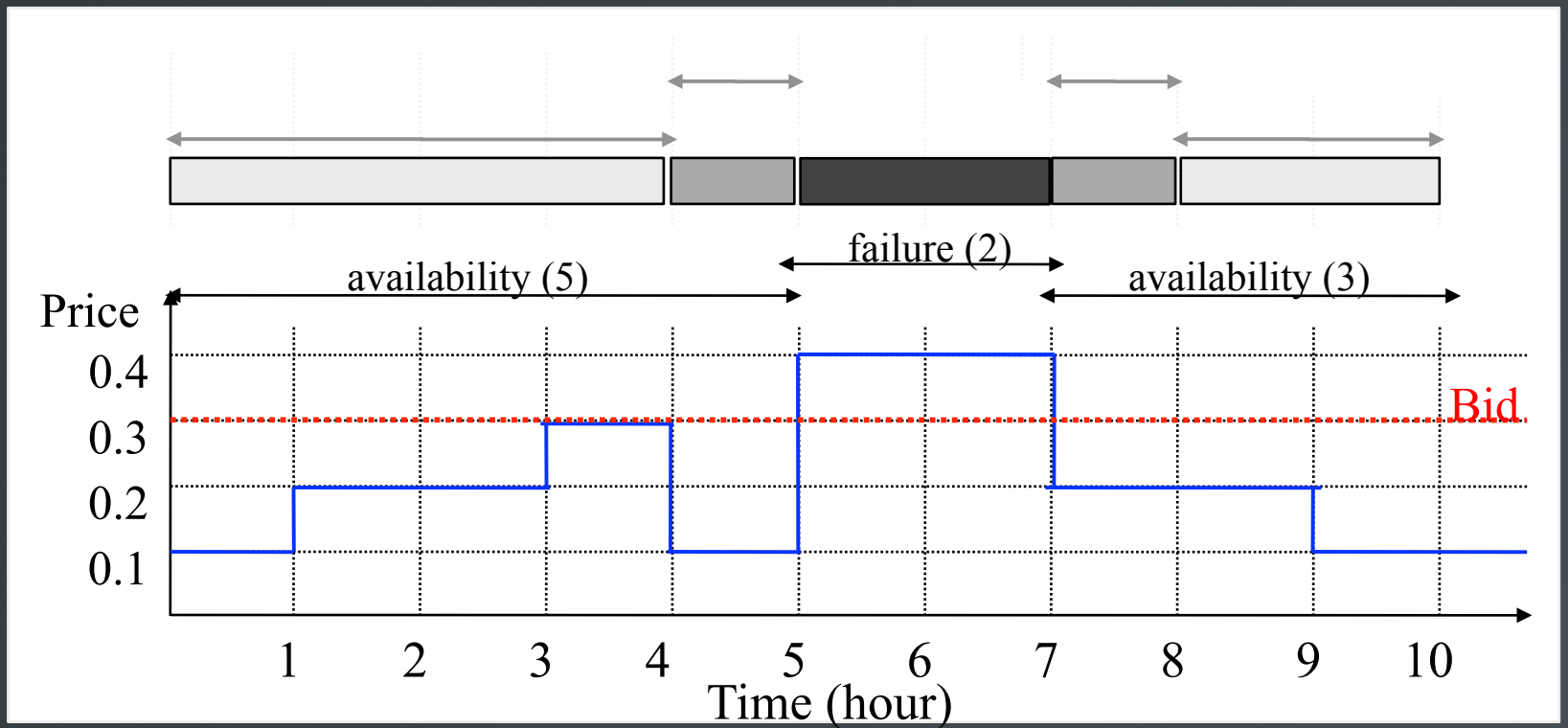


PDF of Failure arrivals



Goal

- Understand **impact of checkpointing** methods on spot instances in terms of **monetary cost** and **execution time**



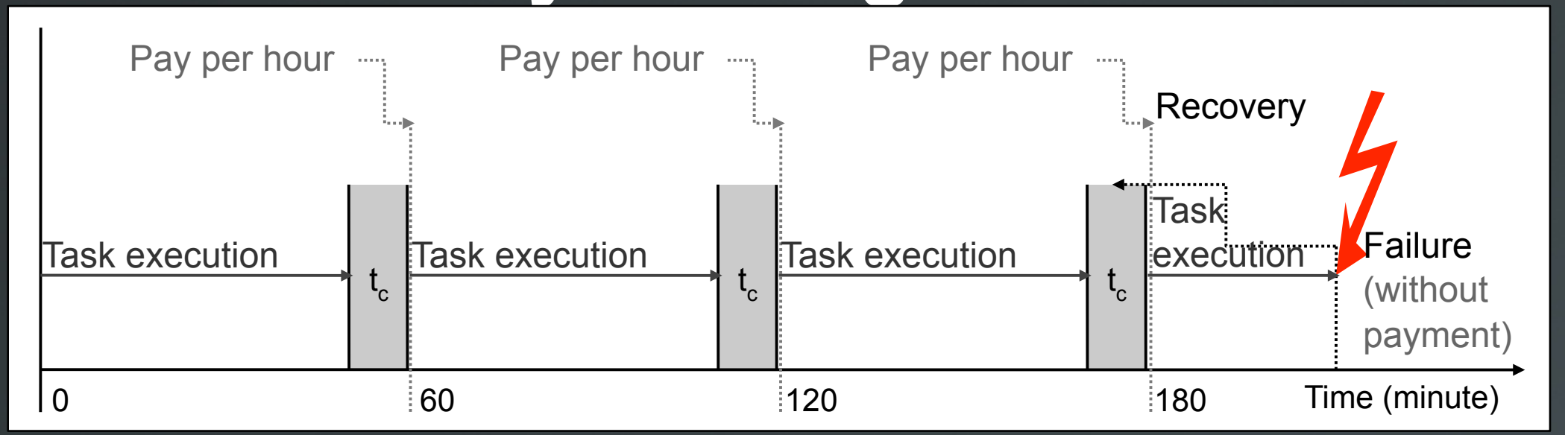
Possible Checkpointing Strategies

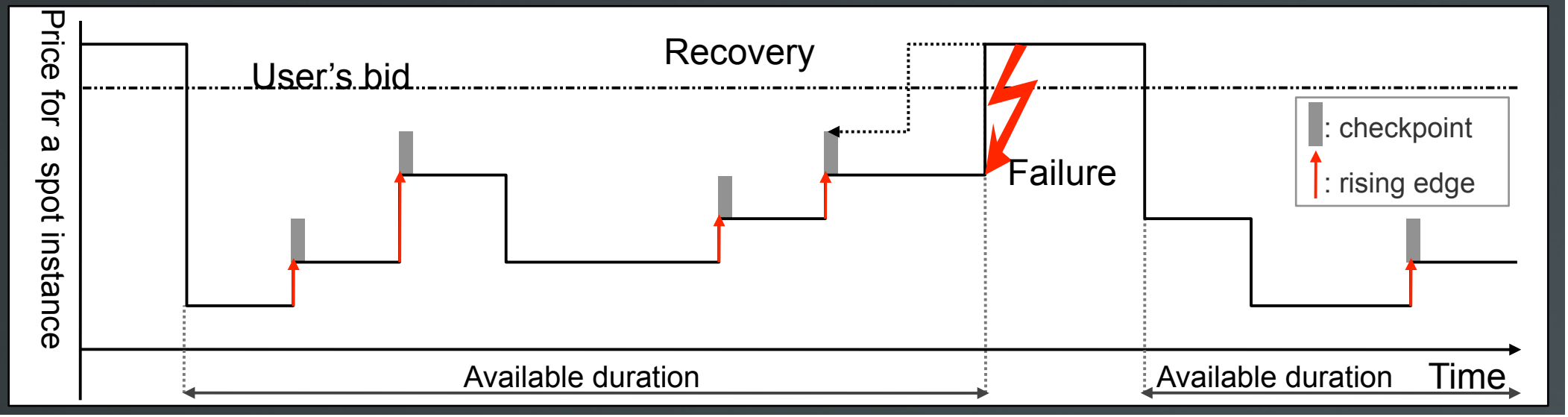
- In terms of taking points
  - The optimal case
  - Without checkpointing
  - Hourly checkpointing
  - Rising edge-driven checkpointing
  - Checkpointing with adaptive decision whether to skip or take a checkpoint
  - Combinations of aboves

Characteristics of Strategies

- Hourly
  - **Pay as you go** (with checkpointing per hour)
- Rising edge-driven
  - **Takes when price goes up** (**except for out-of-bid cases**)
- Adaptive taking point decision
  - **Has overhead for decision**, also **the decision may not work well in some cases**.

Hourly vs. Edge-driven





Adaptive Taking Point Decision

- Based on time series analysis,

it compares the expected costs at a certain point when

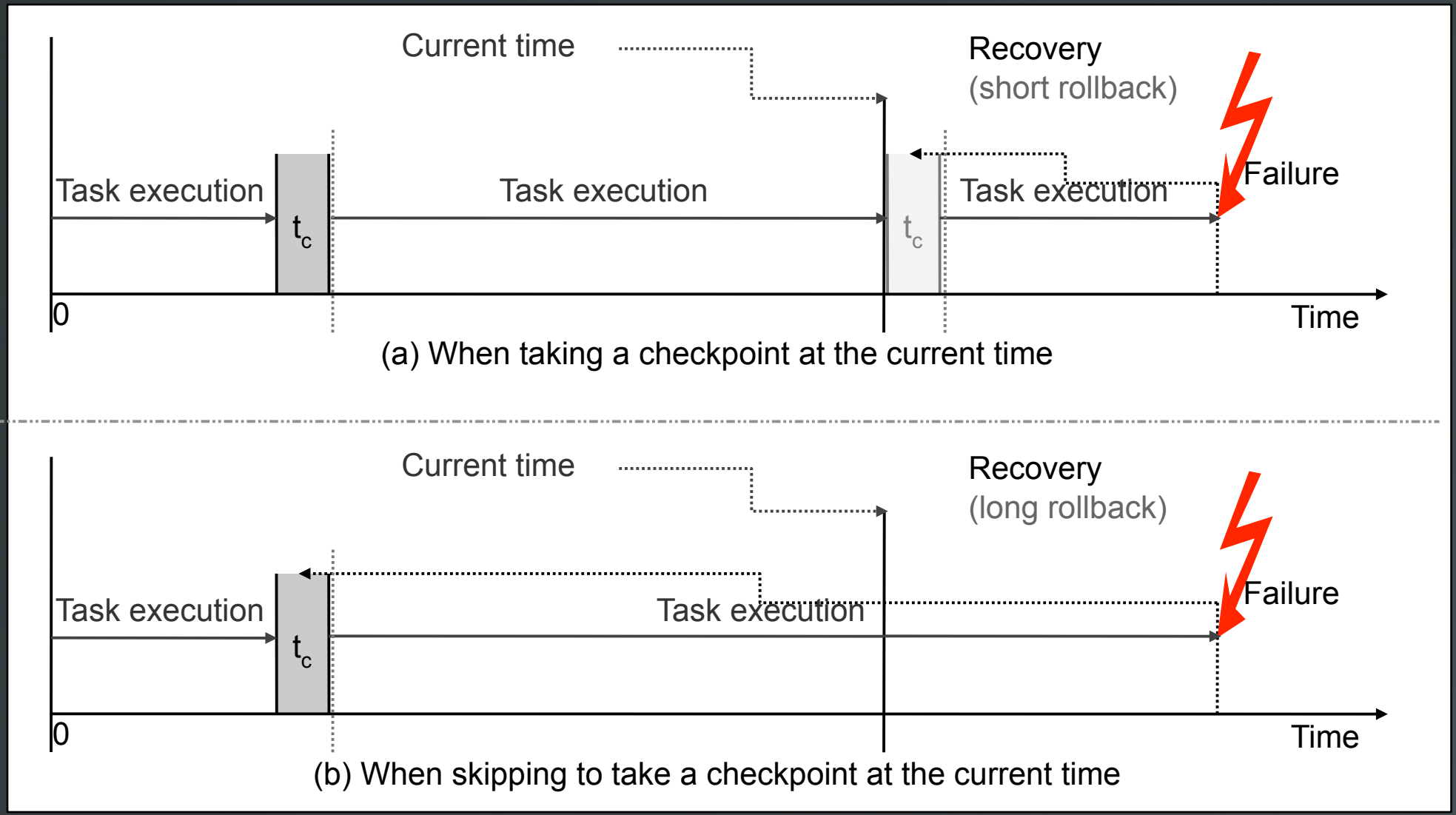
  - Taking a checkpoint
  - Skipping a checkpoint

- Then, selects the lower-cost option.

System Model

Notation	Description
t_r	remaining work requirement time of a task
t_c	time to take a checkpoint
t_a	time to analyze price history for obtaining $f(t, u_b)$
r	time to restart a task
u_b	user's bid on a spot instance type
$f(t, u_b)$	probability density function of failure occurrence where t is time argument, and u_b is user's given bid
$e(t, u_b)$	probability density function of rising edge occurrence where t is time argument, and u_b is user's given bid
n_e	number of arrived rising edges of spot price in the current available interval
$m_e(u_b)$	mean number of rising edges in an available interval according to u_b
$T(t)$	expected execution time of a task without checkpointing when executing t time units
$H_{take}(t)$	expected recovery time of a task with *hour-boundary* checkpointing at t time units after taking checkpoint
$H_{skip}(t)$	expected recovery time of a task without *hour-boundary* checkpointing at t time units after taking checkpoint
$E_{take}(t)$	expected recovery time of a task with *rising edge-driven* checkpointing at t time units after taking checkpoint
$E_{skip}(t)$	expected recovery time of a task without *rising edge-driven* checkpointing at t time units after taking checkpoint

To take or skip, that is the question



Cost analysis on the choice

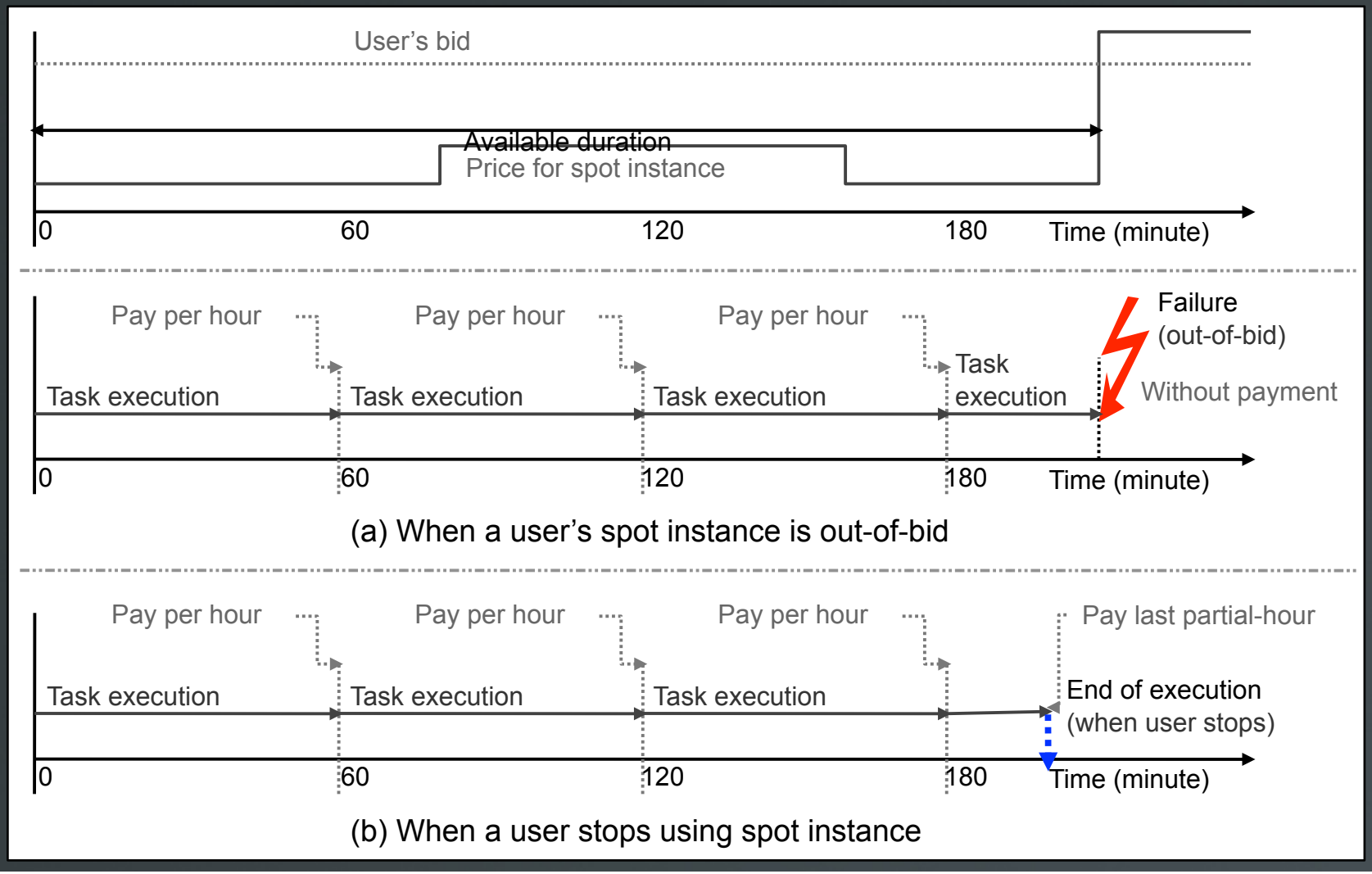
Theorem 2: The expected recovery time when skipping an hour-boundary checkpoint at t time units after taking checkpoint, $H_{skip}(t)$ is given by

$$H_{skip}(t) = \sum_{k=0}^{t_r-1} (k + r + T(t)) f(k, u_b). \tag{2}$$

Theorem 3: The expected recovery time when taking an hour-boundary checkpoint at t time units after taking checkpoint, $H_{take}(t)$ is given by

$$\begin{aligned} H_{take}(t) = & \sum_{k=0}^{t_r-1} (k + r) f(k, u_b) \\ & + \sum_{k=0}^{t_c-1} T(t) f(k, u_b) + T(t_c). \end{aligned} \tag{3}$$

Delayed termination – a way of exploiting Amazon's policy



Performance evaluation

- Trace-based simulation
  - http://spotckpt.sourceforge.net
    - 42 spot instances (now Amazon has 64)
    - 24 (12x2) different checkpointing strategies

Table III
VALUES OF PARAMETERS USED IN THIS PAPER

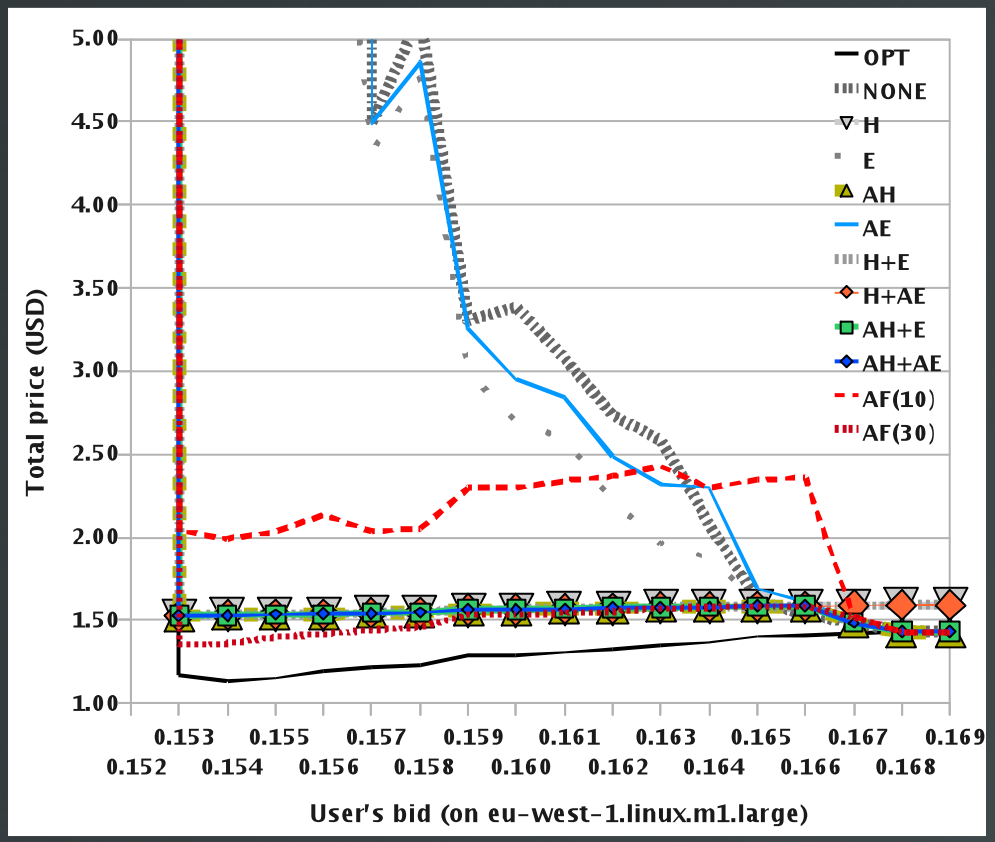
Parameter				
Starting date of past traces			Jan. 11th, 2010	
Ending date of past traces			Feb. 5th, 2010	
Past traces (for calculating pdf)			14,400 mins	
Minimum bidding granularity			0.001 USD	
Parameter	t_r	t_c	t_a	r
Value	500 mins	5 mins	3 secs	10 mins

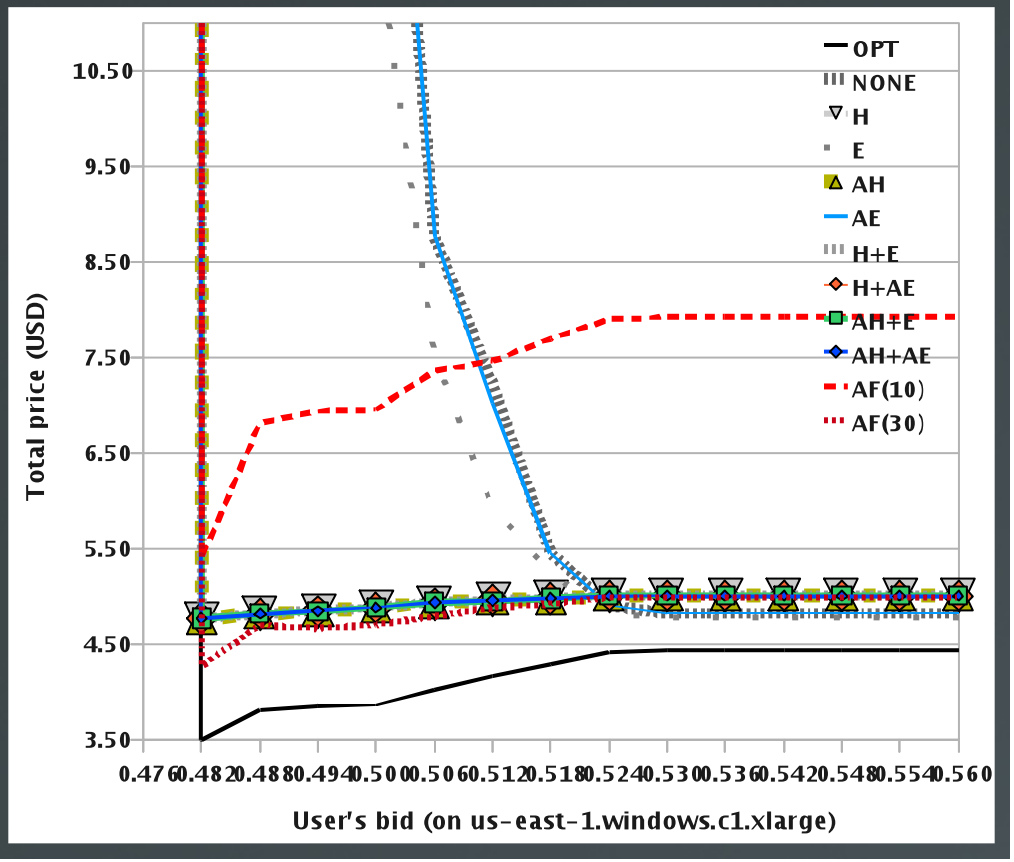
Checkpointing strategies

- OPT: The optimal case
- NONE: Without checkpointing
- H: Hourly
- E: Rising edge-driven
- AH: H with adaptive decision
- AE: E with adaptive decision
- H+E, H+AE, AH+E, AH+AE
- AF(10): adaptive decision every 10 mins
- AF(30): adaptive decision every 30 mins

Results #1: Monetary cost

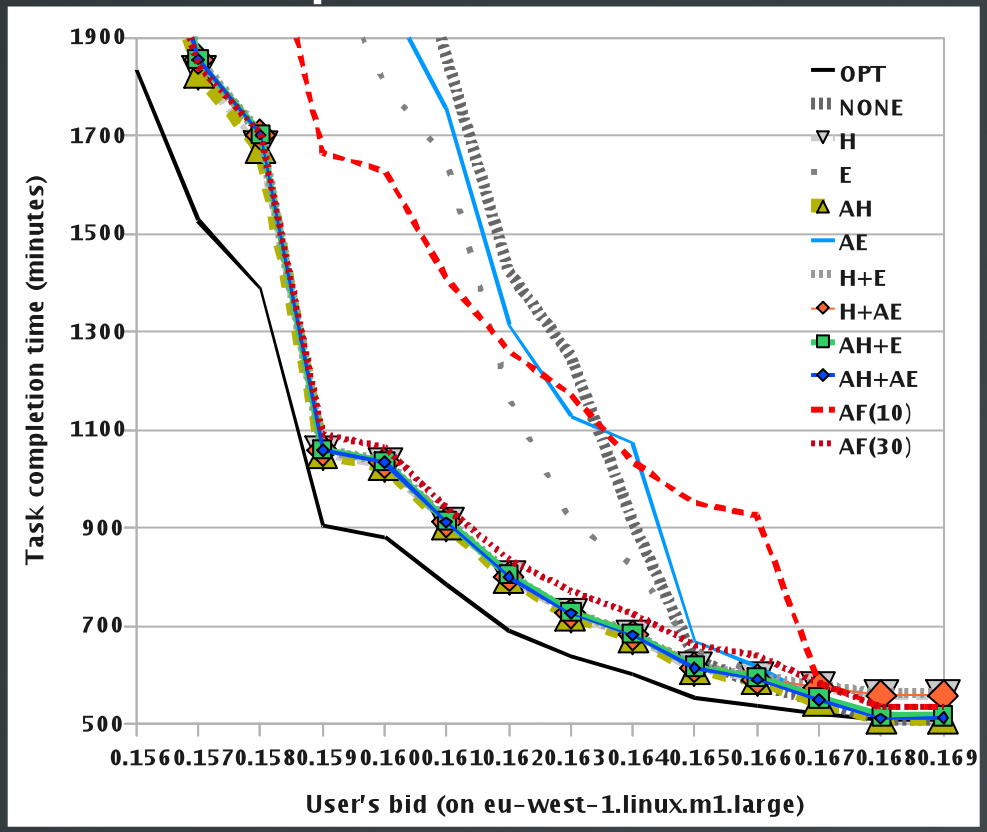
- Total cost according to User's bid on two spot instances

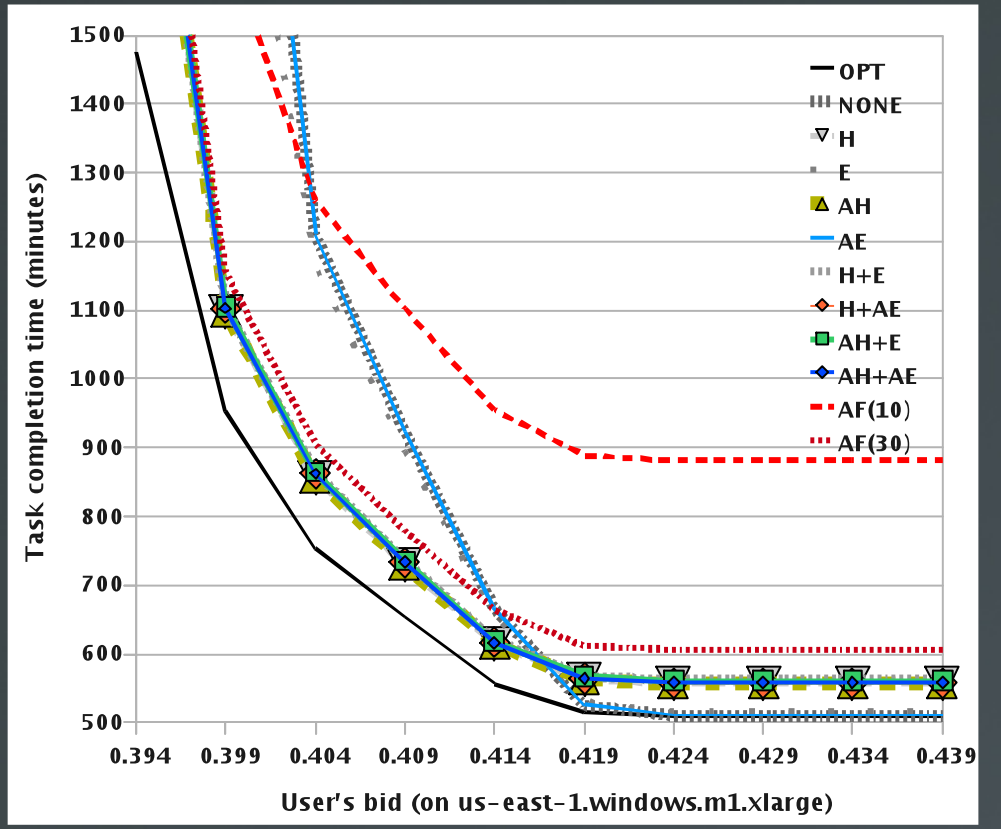




Results #2: Execution time

- Total execution time according to User's bid on the spot instances.





Results #3: Normalized one
Results #4: Benefit of using DT

Table IV
NORMALIZED PRICE×TIME PRODUCT FOR EXECUTION ON THE MEAN PRICE BIDDING (NORMALIZED BY OPT)

eu-west-1.linux type	NONE	H	E	AH	AE	H+E	H+AE	AH+E	AH+AE	AF(10)	AF(30)
c1.medium	2.659	1.298	3.841	**1.296**	2.660	1.307	1.300	1.307	1.300	2.865	1.405
c1.xlarge	19.34	1.454	18.11	**1.450**	32.77	1.460	1.456	1.460	1.456	3.444	1.558
m1.large	6.826	1.408	4.261	**1.405**	5.147	1.420	1.417	1.420	1.417	3.275	1.428
m1.small	16.18	1.505	15.11	1.508	16.18	1.543	1.543	1.543	1.543	2.848	**1.496**
m1.xlarge	13.75	**1.445**	11.50	1.449	16.86	1.448	1.447	1.448	1.447	2.655	1.456
m2.2xlarge	2.894	1.462	2.900	1.462	2.897	1.465	1.464	1.459	1.464	2.690	**1.428**
m2.4xlarge	3.458	**1.354**	2.758	1.355	2.972	1.360	1.358	1.360	1.358	2.843	1.411

Table V
THE AMOUNT OF PRICE REDUCTION (IN USD) WHEN USING DELAYED TERMINATION (ON THE MEAN PRICE BIDDING)

eu-west-1.linux type	OPT	H	E	AH	AE	H+E	H+AE	AH+E	AH+AE	AF(10)	AF(30)
c1.medium	0.021	0.006	0.001	0.006	0.014	0.007	0.006	0.007	0.006	0.003	0.002
c1.xlarge	0.101	0.109	0.067	0.109	0.067	0.105	0.109	0.105	0.109	0.074	0.015
m1.large	0.019	0.029	0.032	0.029	0.043	0.025	0.025	0.025	0.025	0.028	0.008
m1.small	0.004	0.015	0.000	0.015	0.000	0.012	0.012	0.012	0.012	0.008	0.004
m1.xlarge	0.034	0.065	0.275	0.065	0.000	0.065	0.065	0.065	0.065	0.039	0.015
m2.2xlarge	0.033	0.093	0.006	0.093	0.006	0.093	0.093	0.093	0.093	0.049	0.049
m2.4xlarge	0.110	0.257	0.175	0.257	0.175	0.257	0.257	0.268	0.257	0.130	0.022

Results #5: Best ones on the mean price bidding

Table VI
BEST CHECKPOINTING POLICY FOR EACH SPOT INSTANCE TYPE IN AMAZON EC2 (ON THE MEAN PRICE BIDDING, IN TERMS OF PRICE×TIME PRODUCT, EXCEPT FOR OPT)

Instance types	c1.medium type	c1.xlarge type	m1.large type	m1.small type	m1.xlarge type	m2.2xlarge type	m2.4xlarge type
eu-west-1.linux	AH	AH	AH	AF(30)	H	AF(30)	H
eu-west-1.windows	AF(30)	AF(30)	AH	AH	H, AH	H, AH	AH+AE
us-east-1.linux	H, AH	H, AH	AF(30)	H	H, AH	H, AH	AH
us-east-1.windows	AE	H, AH	H, AH	AF(30)	H	AE	E, AE
us-west-1.linux	H, AH	H, AH	AF(30)	AH	H, AH	H, AH	AH
us-west-1.windows	H, AH	AF(30)	AH	H, AH	E	E	AH

- Hourly checkpointing shows good performance.
- Fine-grained checkpointing shows good performance sometimes.
- Adaptive decision does not show significant impact on the results.

Summary

- Checkpointing strategies have ~50% higher overhead (execution time * monetary cost) than the optimal case.
- Thus, better checkpointing strategies are required to reduce both cost and time.

Current and Future work

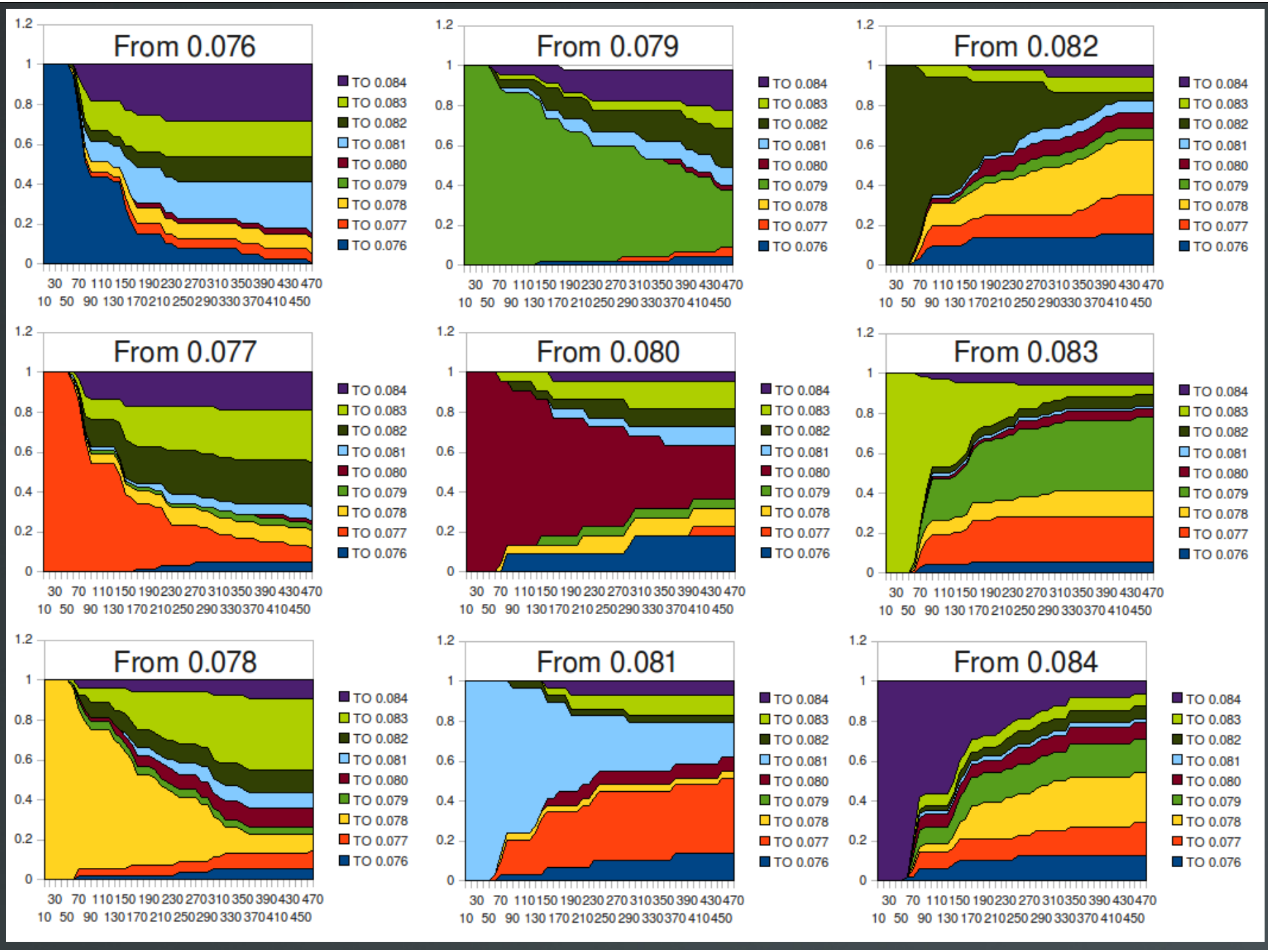
- 1) Finding **relationship** between **past** and **future price** of spot instances
- 2) Developing **efficient checkpointing** method to **minimize monetary cost** and **execution time** (to appear @ EKC 2010)
- 3) Providing **decision model** to determine user's bid, given reliability and performance constraints (to appear @ MASCOTS 2010)

Thanks! / Questions?

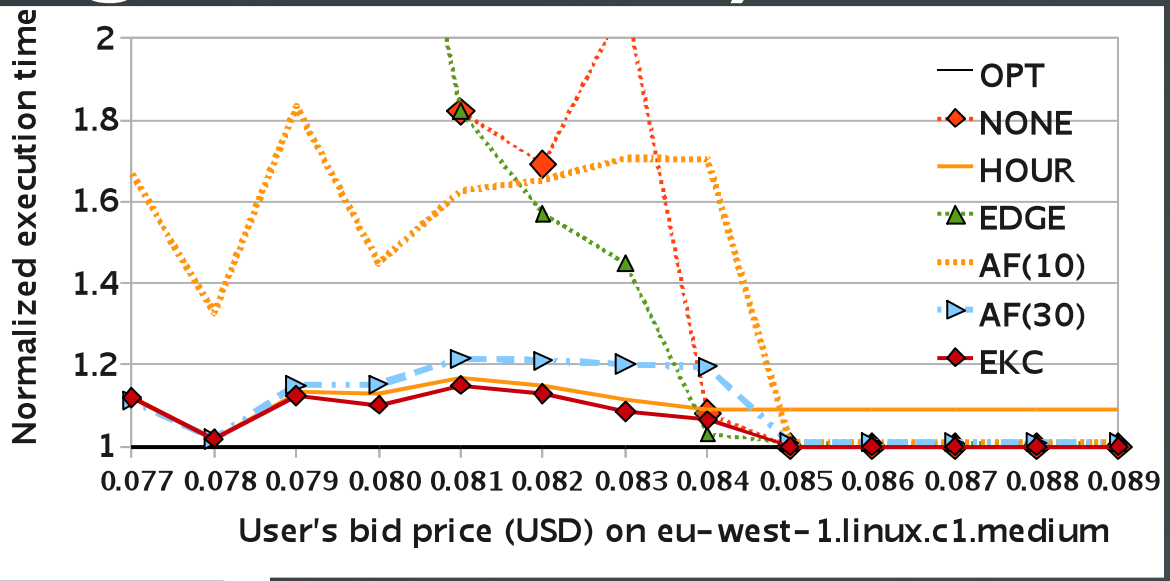
Let’s walk (or work) on the Cloud!

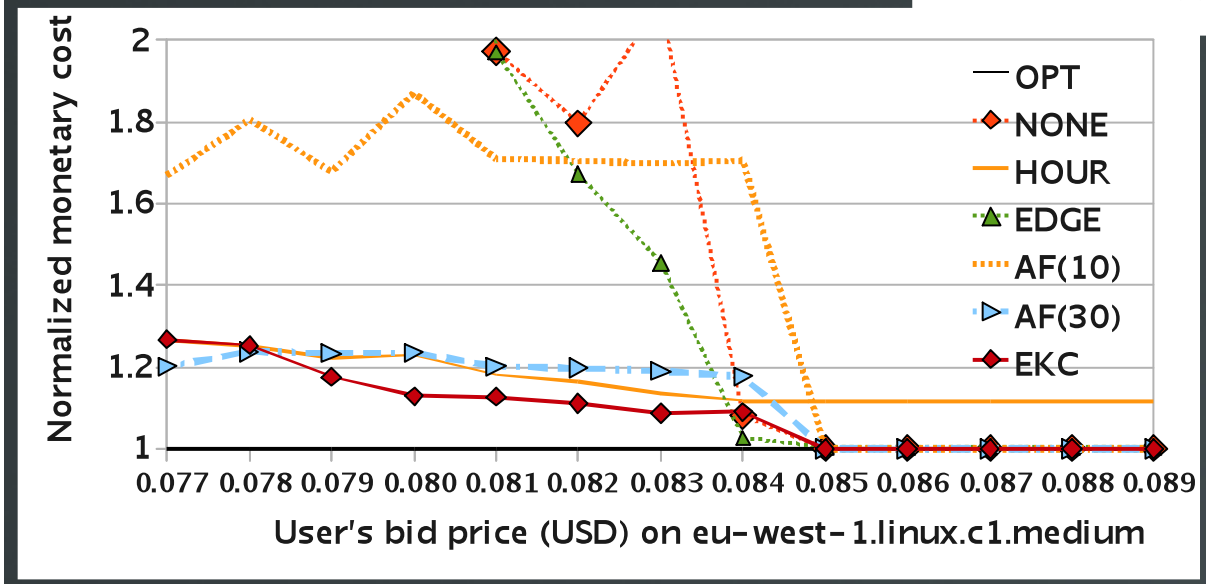
** On-going work on 2)

- Sangho Yi and Derrick Kondo "**How Checkpointing Can Reduce Cost of Using Clouds?**," EKC 2010, July, 2010. (accepted)
  - We used the known characteristics of Amazon's Spot Instances, and better checkpoint decision mechanism.
    - The proposed scheme shows much less overhead for most cases. (~20% overhead)


From 0.076
From 0.079
From 0.082
From 0.077
From 0.080
From 0.083
From 0.078
From 0.081
From 0.084
TO 0.084
TO 0.083
TO 0.082
TO 0.081
TO 0.080
TO 0.079
TO 0.078
TO 0.077
TO 0.076
1.2
1
0.8
0.6
0.4
0.2
0
30 70 110 150 190 230 270 310 350 390 430 470
10 50 90 130 170 210 250 290 330 370 410 450

** On-going work on 2)





Let's Make Our Cloud Better!

- On a Cloud Vendor's point of view,

- TODO
  - (efficiently) Adapt management strategies to fully utilize the limited (very-large) resources (with less cost)